Curriculum Module
SEI-CM-23

Technical Writing for Software Engineers

Linda Levine
Linda H. Pesante
Susan B. Dunkle

November 1991

Carnegie Mellon University
Software Engineering Institute

Technical Writing for Software Engineers

Linda Levine
Linda H. Pesante
Susan B. Dunkle
Carnegie Mellon University

November 1991

This report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

John S. Herman, Capt, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994.

This document is available through Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.
Phone: 1-800-685-6510. FAX: (412) 682-6530.

Copies of this document are available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

This document is also available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Preface

Capsule Description

This module, which was written specifically for software engineers, discusses writing in the context of software engineering. Its focus is on the basic problem-solving activities that underlie effective writing, many of which are similar to those underlying software development. The module draws on related work in a number of disciplines, including rhetorical theory, discourse analysis, linguistics, and document design. It suggests techniques for becoming an effective writer and offers criteria for evaluating writing.

Scope

In defining the scope of this module, we had to make choices within the broad—and growing—field of technical communication. Although we recognize that subjects such as oral communication and group dynamics are important for software engineers, we have set our priority on written communication. Even then, the topic is a large one and difficult to treat in depth. Thus, we have limited the scope of the module to the fundamentals of the writing process.

The module does not include material on business writing (memos, proposals, etc.), oral presentations, group dynamics, or project management. Although technology (from basic word processors to hypertext and desktop publishing systems) certainly has great impact on how people write, we cannot do justice to the subject within the limits of this module. Nor do we devote space to the characteristics of specific types of documents and other information, such as grammar rules, that can easily be found in other sources. References to these sources appear in the bibliography.

Because we concentrate on the basics, the material in this module applies to many types of documents, and some of the material can be applied to oral presentations as well. The module thus provides the foundation for studying other communication topics and building further skills.

Philosophy

Need for communication skills in engineering

Communication skills are important in every engineering discipline. Surveys by organizations such as the American Society for Engineering Education indicate just how critical these skills are [Olsen91, Barnum84]. Those who are not convinced should consult the 20 references provided at the end of Chapter 1 in [Olsen91]. These references discuss the place of writing in engineering. For specific information on software engineers' need for communication skills, see [Sullivan88, White86, Curtis88, Guinan87].

Assumptions about writing

Ineffective writers—engineers included—often feel that writing should be easy even though that has rarely been their experience. They may justify their difficulty in a number of ways; for instance, unskilled writers suspect that their teachers were inadequate, or they blame themselves for just not having "the gift," or both. But gifts, muses, and favorite pencils aside, conceptions of writing are finally changing. After decades of instruction in writing and centuries of instruction in its precursor, rhetoric, researchers are beginning to tell us about the social and cognitive complexity of the writing process.

This research has also revealed problematic and lingering assumptions and "myths" about writing. Two assumptions about the writing process are especially significant. The first views writing as an "art" and carries important consequences for educators, who then sustain the mystique of that art and see themselves as facilitators more than teachers. This approach, which assumes that writing is a gift that can't be taught, often translates into a view of writing as discovery of one's own inner voice [Young80]. A second assumption about the writing process sees thinking and writing as separate activities. Here, thinking is accorded the prestige of an art or a science, and writing is seen as the "craftlike" translation of these ideas into words. Related to this notion of writing as translation is a view of writing as editing. The most limited definition calls writing the simple polishing of words, or "wordsmithing."

Two myths concerning the written product are especially prevalent in technical communication. First is the myth that technical writing is transparent, objective, and fact-based writing. (Carolyn Miller's treatment of this "windowpane theory of language" as a legacy of positivism is worth noting [Miller79].) Technical documents are designed to report information clearly and persuasively, and an objective tone should not be confused with objectivity. Finally, there is the widespread belief that following rules and/or formulas will guarantee a good product. However, a piece of writing can be error-free and still not communicate effectively. Tichy, Kirkman, and Young all discuss these myths about writing in greater detail, while Hoare and Weizenbaum consider related myths about computers and programming [Tichy88, Kirkman70, Young80, Hoare84, Weizenbaum88].

There are no algorithms for writing. Writing is rhetorical, requiring the writer to take into consideration many factors that change from one situation to the next. Because writing is not rule-driven, this module provides strategies or heuristics—not rules—that will help software engineers become more effective, more self-aware writers. These heuristics are based on research from a number of disciplines, including psychology, rhetoric, linguistics, discourse analysis, and document design.

Our approach to writing

We believe that writing should be taught within software engineering content courses, not in a separate, single course. If a separate technical communication course is offered as a foundation, it must be tailored to meet the writing objectives and needs of software engineers; for example, it should be taught in conjunction with a software project for another course. Students should write the kind of documents they will write as software professionals.

Research tells us that students have difficulty applying writing skills learned in the composition or technical writing classroom to writing in different domains. If skills learned in the writing classroom aren't applied in the software engineering classroom, it is unlikely that the new engineer will draw on those skills in the workplace. Moreover, effective writing is not something that can be covered once and mastered. To address these problems, we stress the need to integrate writing into the curriculum, making it a part of each course that students take.

Our approach to writing is not art, not science, not craft, but more in the tradition of design (see Section 3 of the annotated outline). Writing is an analytical activity. We believe that writing skills will help individuals to be better engineers, and engineering skills will help individuals to write better. Both writing and software development are problem-solving processes requiring practitioners to perform similar tasks; therefore, in this module, we look at the similarities between these processes. Teaching by analogy allows us to efficiently exploit the correspondences between the processes of writing and software development. Learning by analogy provides our students with a powerful mechanism for applying the skills they already have as computer scientists and software engineers. Although there is not a dovetail fit between the two, it is useful to exploit the strong parallels.

Our focus on process, rather than on specific types of documents or on rules, also facilitates the *transfer* that we see as so important. With an understanding of the writing process, of the cognitive components that are a part of that process, and of rhetorical situations, software engineers will be able to communicate effectively in the wide variety of documents that software development projects require.

While our approach is meant for application in the software community, the growing number of writing-across-the-curriculum projects also lends support for our position. The interest in these projects demonstrates that poor communication, especially ineffective writing, is being recognized as a shared concern that involves all departments, not just the English department. All instructors teach communications skills whether or not they are aware of it. All instructors can and should evaluate their students' writing—both their written products and their processes. Those who don't are doing their students a real-world disservice. These students will soon be accountable to employers (not English instructors) who will evaluate their ability both to perform and to communicate.

Acknowledgements

The authors would like to thank the members of the SEI Education Program, the technical writers of SEI Information Management, and the SEI librarians for their assistance. In addition, we thank Michael Rissman and Daniel Klein for providing technical perspective. We are especially grateful to our reviewers for their insights and valuable suggestions: Thomas Huckin, Granville “Pete” Jones, Patricia Lawlis, Richard Rasala, and Rachel Spilka.

Comments on this module are solicited, and may be sent to the SEI Software Engineering Curriculum Project or to the authors:

Linda Hutz Pesante
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Susan B. Dunkle
401 Warner Hall
Carnegie Mellon University
Pittsburgh, PA 15213

Technical Writing for Software Engineers

Outline

1 The Context for Writing

1.1 Rhetorical Situation

1.2 Communication Triangle

1.3 The Aims of Discourse

1.4 Disciplinary Context

1.4.1 Disciplinary knowledge

1.4.2 Discourse communities and conventions

2 Views of Writing

2.1 Product-Based and Process-Based Views

2.2 Models

2.2.1 Communication models

2.2.2 Writing models

3 Analogies: Software Development and Composing

3.1 Art/Science/Design

3.2 General Correspondences Between the Disciplines

3.3 Specific Analogies: Products and Processes

3.3.1 Products

3.3.2 Processes

4 The Writing Process

4.1 Analyzing

4.1.1 Problem definition

4.1.2 Task definition

4.1.3 Audience analysis

4.2 Planning

4.2.1 Product plan

4.2.2 Process plan

4.3 Generating text

4.3.1 Rapid prototyping

4.3.2 Stepwise refinement and iterative enhancement

4.3.3 Risk-driven approach

4.3.4 Reuse

4.4 Testing

4.4.1 Informal testing

4.4.2 Formal testing

4.5 Revising

4.6 Maintaining

5 The Written Product

5.1 Principles of Linguistics and Discourse Analysis

5.1.1 Global concerns

5.1.2 Local concerns

6 Other Considerations

6.1 Collaborative Writing

6.2 Document Design

1 The Context for Writing

Annotated Outline

1.1 Rhetorical Situation

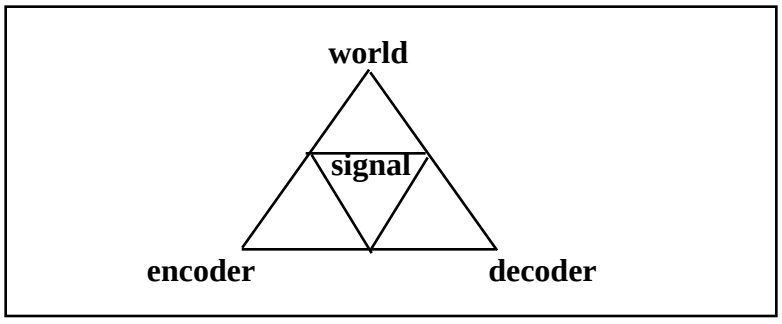
Writing does not take place in isolation but in specific contexts; moreover, writers and readers of a document are often in different circumstances, with different goals, constraints, and conflicts. The context in which communication takes place is called the *rhetorical situation,* which includes writers, readers, and their purposes for writing and for reading [Young70]. Other elements include the discourse community being addressed (see Section 1.4.2) and the conventions for the kind of document being written. Writers base their decisions about content and language on their understanding of the rhetorical situation. Those who do not take the rhetorical situation into consideration are unlikely to be effective communicators.

For a more detailed, theoretical discussion of the rhetorical situation, see [Bitzer68]. Bitzer identifies three components that are essential to the rhetorical situation: the writer's *exigence* (or sense of an imperfection or problem that needs to be addressed), the audience, and the constraints. Constraints include what are called *artistic proofs* that the writer can manage (e.g., lines of reasoning, word choice, organization) and *inartistic proofs* (e.g., contracts, agreements, standards) that the writer cannot control.

Teaching Consideration: Technical topics that provide a good opportunity to discuss the rhetorical situation include requirements engineering and user interface development. SEI curriculum modules on these topics explicitly discuss issues of audience [Brackett90, Perlman89].

1.2 Communication Triangle

The communication triangle, or rhetorical triangle, presents the essential elements in communication [Kinneavy71].



The *encoder* (writer or speaker), the *decoder* (reader or listener), and *world* (subject matter and environment) each lie at a tip of the triangle. In the center is the language or other *signal* (mode of communication). The writer and reader interact with each other and with the subject matter; the language touches all the other elements in the triangle. These elements and their interactions play a part in any communication, oral or written. See Section 2.2 and Section 4.1 for further discussion.

Teaching Consideration: It is important to note that the communication triangle does not explicitly recognize the place of purpose in language use—purpose of the writer and the reader. This is the reason we present information about the rhetorical situation as well as the communication triangle. If students use the communication triangle as a model, you might need to remind them to consider their purpose for writing and their readers' purpose for reading.

1.3 The Aims of Discourse

Kinneavy categorizes forms of discourse and the purposes for writing each type by considering which element in the communication triangle dominates. These categories are not exclusive because all the elements are involved in each type of discourse; nor are purposes as well defined as the following list may suggest. For example, poetry calls attention to the language, but the poet may also want to achieve originality and persuade the reader of a particular point of view—poetry is literary discourse, but it contains elements of expressive and persuasive discourse. Similarly, a proposal must be persuasive, but it also contains technical data; a technical report contains factual information, but it reflects the choices and assumptions of the writer [Kinneavy71].

- Literary discourse: language is the main concern, as in poetry. The main purpose is to induce contemplation and enjoyment and to call attention to language itself.
- Expressive discourse: the writer is the main focus, as in diaries and journals. The main purpose is to achieve originality or to make the writer's thoughts or feelings accessible to himself or herself.
- Persuasive discourse: the reader is the main focus, as in proposals and essays. The main purpose for writing is to persuade, that is, to modify the attitudes or behavior of the reader.
- Referential discourse: the subject matter is the main focus, as in scientific and technical writing and news reporting. The main purpose is to provide information about aspects of the

world. Documents that support the software life cycle are forms of referential discourse.

In scientific writing, comprehensiveness is seen to be the chief characteristic, as in records of scientific research. In technical writing, factuality is considered the primary concern; writers select information to meet the needs of the audience.

1.4 Disciplinary Context

Knowledge of a discipline and discourse communities are primary issues in communication. The ways people communicate and their effectiveness are influenced by these issues.

1.4.1 Disciplinary knowledge

The issue of what constitutes disciplinary knowledge is a new area of research in rhetorical studies. This effort represents a need to understand the inner workings of different disciplines or fields of study. The goal is to make the theories, methods, and practices of those fields explicit; once they are explicit, they can be disseminated. For discussions of the rhetoric of science and technical writing, see [Halloran78, Miller79].

1.4.2 Discourse communities and conventions

Those who take a social, outer-directed, approach to composing emphasize discourse (or interpretive) communities. This approach focuses on how members of professional and academic groups share patterns of reasoning and language use [Bizzell82, Bruffee84, Odell85]. The term used to describe these shared and accepted practices is *discourse conventions*.

Discourse conventions bind and guide members' interactions in professional and academic communities. Such conventions include: research methods of inquiry and investigation, modes of proof, commonly held assumptions, conference and publication codes, and standards. It is these conventions that novices adopt when joining a discourse community, for example, when students learn to think and behave like software engineers. Doheny-Farina focuses on an interesting double interaction between convention and community: he studies how "social" and organizational contexts affect the writing of a business plan and how the writing of that plan affects the organization [DohenyFarina86].

1.4.2.1 Standards

The requirement to write to standards, such as IEEE standards or military standards, is an excellent example of a discourse convention that imposes constraints. Although standards constrain writing and should be considered during analysis (Section 4.1), they do not eliminate rhetorical choices or decisions. By paying attention to audience, purpose, and functional principles of linguistics (Section 5), a writer can write to standards *and* write for readers. Two documents with roughly the same content may be written to a standard, yet one may be more readable than the other [Penrose88].

1.4.2.2 Plain English

One movement that has grown up in response to problematic discourse conventions is the Plain English movement. This movement (which gained momentum when Plain English laws were passed during the Carter administration) originally focused on rewriting consumer documents such as insurance policies, loan agreements, government publications, and instructions for using products. The influence of the movement has spread to technical writing, where stress is being placed on simplicity, clarity, and judicious use of jargon. This stress is evident in most technical writing books, including [Olsen91, Tichy88, Williams89].

2 Views of Writing

2.1 Product-Based and Process-Based Views

Two common approaches to writing are based on products and processes.

Product-based: This approach focuses on grammar and style and is still a stronghold in writing education. The perspective is rule-based, relying on formulas and model texts for the writer to imitate. Product-based views assume that there are clear right and wrong answers in language usage, that if a writer imitates an effective sample document, the imitation will be as good as the sample.

Guidebooks on grammar and punctuation are useful, but they neglect writing activities (analysis, planning, and others; see Section 4) that must occur before the product can be examined at the level of these mechanical details. Similarly, models provide helpful information about how certain documents should look (see Section 1.4.2), but they do not take rhetorical situations into account. Williams offers advice on rules: why they exist, when to follow them, and when not to [Williams89]. Miller has an interesting discussion of rules and their source of authority [Miller80].

Process-based: This approach focuses on how individuals compose, and it draws on related studies in cognitive psychology, especially problem solving. Researchers have used think-aloud protocols of individuals performing writing tasks in order to describe the subprocesses writers engage in. Using information from the protocols, researchers identify strategies used by expert and novice writers. These strategies are adapted for use as learning tools in writing classes. This approach assumes that writers can become more effective by monitoring their writing processes, and extending and remaining aware of their options [Flower89, Perl80, Selzer83, Sommers80]. For further discussion of the writing process as a design activity, see Section 3.1 and Section 3.2.

Each approach has limitations, and each makes a contribution to understanding writing. Recently, critics have looked at how an approach that addresses the social and cognitive aspects of composing would mediate between product- and process-based methods [Bizzell82, Bruffee84].

2.2 Models

The models identified below are only a small sample. The communication triangle, a model that takes a broad view of the factors involved in communication, is described in Section 1.2. Other models are discussed in the source texts cited in this section.

2.2.1 Communication models

Communication models apply to all forms of communication, not just the written form. Standard communication models show a linear progression from one phase to the next. The Shannon-Weaver model illustrates how information moves from a source through a transmitter and a receiver to a destination. At midpoint, the model accounts for “noise” or distortion. The Lasswell model handles acts of communication by posing questions: *Who? Says what? In which channel? To whom? With what effect?* Both models appear in [Schutte83, Kinneavy71].

More recently, Rogers and Kincaid have developed the convergence model of communication and network analysis. This nonlinear model emphasizes the processes of expression and interpretation: participants are not seen as encoders or decoders; they assume the roles of both transmitter and receiver—as *transceivers*. "Communication is defined as a process in which the participants create and share information with one another in order to reach a mutual understanding" [Rogers81].

2.2.2 Writing models

Prescriptive: These models of the writing process resemble the waterfall phase model and show how writing proceeds through an orderly sequence of stages. Linear models (sometimes called stage models) show prewriting first, then writing, and then revision. The model in [Rohman65], which identifies these three stages, is a typical example.

Descriptive: The document design model prepared by the American Institutes for Research (AIR) is a more descriptive model, treating the general process for all document production [Schutte83, Redish89]. However, some critics have noted that the model seems prescriptive in its linear presentation of predesign, design, and postdesign steps.

Flower and Hayes present a model of composing that describes the processes and subprocesses that writers engage in [Flower81]. Based on their research with individuals performing think-aloud protocols while writing, the authors distinguish three main components: the writer's long-term memory, the task environment, and writing processes. The authors identify subareas and subprocesses within these main components. They see writing as a recursive process because "this particular kind of embedding, in which an entire process is embedded within a larger instance of itself, is known technically in linguistics as recursion." In another paper [Hayes87], the authors develop a model to represent cognitive processes in revision. Major subprocesses in this model are: task definition, evaluation, problem detection, problem diagnosis, and strategy selection.

Currently, researchers in rhetorical studies are trying to develop models that account for, and describe, the social and cognitive dimensions of the composing process [Bizzell82, Bruffee84].

3 Analogies: Software Development and Composing

This section discusses analogies that have been made between software engineering and writing. It is an introduction to a growing body of literature that explores the similarities between the two domains. The first subsection describes a dialogue common to both fields, one that considers these disciplines as art, science, and design. The second notes general correspondences between the fields of software engineering and writing; and the final subsection discusses specific analogies.

This presentation of analogies is not exhaustive. It aims, rather, to highlight key concerns that have been most frequently addressed in the literature. Readers will find additional similarities between software development and composing in these and other sources.

3.1 Art/Science/Design

Ongoing discussions about whether software engineering and writing are arts or sciences support the consideration of both fields as design disciplines. The concept of design accounts for artistic performance, giftedness (or wizardry), as well as the use of scientific methods to investigate and explain writing and programming. Walton and Balestri point out the advantages of using Herbert Simon's sense of design as problem solving [Walton87]. Simon notes that engineers, like designers, are "concerned with how things *ought* to be—how they ought to be in order to *attain goals*, and to *function*." According to Simon, "natural science is primarily descriptive, considering things as they are; the sciences of the artificial (the man-made) are normative, concerned with how things should be" [Simon81].

3.2 General Correspondences Between the Disciplines

While the discussion about the art and science of software engineering and of writing can be mediated through the concepts of design and problem solving, disagreements about the issues are common to both disciplines [Hoare84, Weizenbaum88, Young80]. Frequently, the disciplines are also compared with one another. Shore describes programming as a literary activity, as mathematics, and as architecture [Shore85]. Walton and Balestri consider both composition and programming as design disciplines and discuss how the process of structured programming resembles the top-down and goal-directed process for writing purposeful prose for a target audience [Walton87]. Lehman compares program design and rhetoric and maintains that from an information

processing point of view, problems in rhetoric are similar to problems in programming: "[R]hetorical solutions to these problems resemble the major tenets of structured programming and structured design" [Lehman86].

Other critics discuss analogies in terms of approach, method, and style. For example, in *The Elements of Programming Style*, Kernighan and Plauger note that the form and approach of their book has been strongly influenced by *The Elements of Style* by Strunk and White [Kernighan74, Strunk79]. Donald Knuth develops the term literate programming to stress the connections between natural and computer languages and the need for individuals to be able to read programs [Knuth84, Bentley86a]. Gary Perlman extends this metaphor through multilingual programming, a process to coordinate program implementation with "the use of parameterized information for domains outside programming, like documentation and user interfaces" [Perlman86].

Software engineering and writing share the debate about their status as art or science (and the discussions about genius or wizardry growing out of this debate), related methods or processes, and surrounding popular fears. In *Orality and Literacy: The Technologizing of the Word*, Walter Ong notes that reservations about the inhumanity of computers match Plato's objections to writing: that it is artificial, unresponsive, and weakens the mind by destroying memory. Ong also notes that with literacy, the word became increasingly a visual unit and less an auditory sound [Ong82]. Hamlet observes a similar and related danger and extends the argument: "documents produced with computer assistance are often of lower quality than those produced by hand: they *look* beautiful, but the content and organization suffer." To correct these problems, he proposes a "disciplined text environment" through the use of stepwise refinement and iterative enhancement methods and rule-based editors [Hamlet86].

3.3 Specific Analogies: Products and Processes

3.3.1 Products

Specific analogies between software engineering and writing usually concern the processes of development. However, in some instances, analogies can also be made between products. For example, Walton and Balestri observe the structural similarities

between the writer's outline and the programmer's Warnier-Orr diagram [Walton87]. The hierarchical, top-down nature of issue trees, a modification of the traditional outline, has a still stronger resemblance to the Warnier-Or r design tool [Flower89].

Product-based analogies relate to process issues when, for example, emphasis is placed on the outlining technique, not the outline. Likewise, software life cycle models and document design models are products that reveal similarities about how researchers (either prescriptively or descriptively) represent the processes of development. Hamlet's discussion of analogies between formatting programs and assemblers is another fine example of product/process relations. He discusses users' inability to regulate word processing capabilities and the resulting problems in local format details and document content [Hamlet86].

3.3.2 Processes

Of the software development methods adaptable to document development, stepwise refinement and iterative enhancement and prototyping have received the most attention. Separation of concerns and information hiding have also been discussed. (See also Section 4.3 and [Levine91].)

3.3.2.1 Stepwise refinement and iterative enhancement

Mingione notes that "documentation must be iterative to serve as systems communications within the development process." In his view, iteration is an easier way to document because it is a "natural" fundamental concept in development [Mingione83]. Hamlet suggests that "techniques and tools valuable in controlling software development be investigated for improving document preparation." He considers top-down iterative enhancement and stepwise refinement, and environments controlled by rule-based editors, as the most promising. Iterative enhancement and stepwise refinement are complementary methods that guide modifications of, and additions to, the hierarchy used for the prototype [Hamlet86].

3.3.2.2 Prototyping

Guillemette's comprehensive discussion of prototyping as a method for developing documentation also notes the place of iterative design within in the process. He identifies four steps, and a finite series of

implementation/revision cycles, in the prototyping approach: (1) completion of needs analysis and development of baseline documentation required for discussion and iteration, (2) development of working documentation, (3) release of baseline documentation for testing, (4) revision based on reader/user feedback. Steps 3 and 4 are repeated until requirements are established [Guillemette87]. In addition, user feedback may indicate flaws in the needs analysis of Step 1. Rapid prototyping techniques are "techniques for constructing working models of systems rapidly and cheaply." Taylor and Standish see this as an appropriate learning method when the ends or the means of system requirements are unclear, or when there are changing requirements. The rapid development of initial versions of a system is a useful analogy (and strategy) for developing initial versions of a document [Taylor82].

3.3.2.3 Separation of concerns

In [Hester81], the authors identify separation of concerns and information hiding as key principles in their design and documentation method. Both principles describe the same essential idea from two perspectives—what the authors call encapsulating *all* elements of *each* aspect and, hence, hiding the information about each aspect. "The principle of separation of concerns is used to structure the design documentation, and information hiding is used to guide the internal design of the software." The authors discuss the design considerations associated with each step that the document covers. They also provide guidelines for the preparation of these documents.

4 The Writing Process

The following sections outline major activities in the writing process. Although the outline must present these activities in a linear way, writers rarely perform them linearly.

4.1 Analyzing

4.1.1 Problem definition

Problem definition involves identifying the problem the document will address or the need it will fulfill. For example, in analyzing the problem addressed in a technology transition plan, the writer describes how a specific piece of new technology will be moved into commercial development, how it will solve a particular problem, and how it will fulfill the needs of a particular organization. In order to design an effective transition plan, or any other document, writers must identify the need for the document and explain how the document meets that need for both themselves and their readers.

In a document, the problem definition sometimes takes the form of a traditional problem statement; at other times, it takes the form of a simpler purpose statement [Young70, Mattes76, Olsen91]. A formal problem statement is most often used when the writer determines that the reader must understand the larger, organizational context as well as the technical issue being addressed. This statement usually includes the following: description of the problem, questions arising out of the problem, and what the document is designed to do in response to the problem. In contrast, a purpose statement simply lets the reader know the purpose of document, without the elaborate context; for example, “this manual describes how to . . . ,” “this memo authorizes. . . .”

Both types of statements frequently appear at the beginning of documents because readers need to know up front what the document is designed to do and why the writer has written it. Readers should not have to infer what the writer intends to accomplish; if the writer is explicit, readers can more easily process the information and are less likely to misunderstand. The circumstances of use will affect content decisions as well as design and production decisions, including the size of pages and type of binding.

4.1.2 Task definition

Writers begin defining their task by identifying their goals and the constraints under which they will be working. Goals, which are incorporated into the plan (see Section 4.2), include identifying what must happen before writing can begin and during the writing process as well as what the writer wants to achieve with the finished text. Constraints include special characteristics of the audience (see below), time, budget, the technology, or the requirement to write to a standard.

In addition, the writer and reader each brings previous knowledge, which affects the writer's content choices and the reader's ability to assimilate the new information that is presented. Writers need to identify the conventions and assumptions that are common, though perhaps unarticulated, between themselves and their readers. Certain conventions are appropriate—and expected—for various types of documents and discourse (see Section 1.3). These conventions affect content, organization, word choice, level of formality, and format. For example, a journal article does not look like a memo; a user's manual does not look like a requirements document. (See also Section 1.4.2.)

If the document is one of a set, the writer has another concern: How does the current document relate to the others in the family of documents? Writers also need to identify criteria for evaluating whether they succeed at their task. These criteria can then be applied during evaluation and testing. For more on testing, see Section 4.4.

4.1.3 Audience analysis

Writers begin analyzing audience by identifying their primary reader(s) and determining what other readers might use the document. They need to analyze the readers' purpose for reading and to identify the context in which the document will be used.

When writers create a document for their peers (the other members of their project team, for example), their audience analysis is likely to be quite accurate, particularly since ongoing interaction enables writers to refine their understanding of their audience. However, it is more difficult to identify the characteristics and needs of a less familiar audience, or multiple (mixed) audiences. Useful information about audience analysis appears in [Olsen91, Schutte83, Roundy85]. Spilka specifically addresses strategies for analyzing multiple audiences [Spilka88].

Teaching Consideration: Beware of texts that present elaborate audience analysis guidelines with detailed checklists. This type of analysis consumes a lot of time and doesn't tell the writer how to apply the results when writing. Analyses that tell the writer how to apply the results are more desirable; these are often the simpler methods of analysis (see [Andrews82b, Olsen91]).

4.2 Planning

Based on their analyses, writers can develop two kinds of plans: product plans and process plans. Because writing is not a linear activity, writers are likely to do further analysis during the writing process, making it necessary for them to revise their plans at later points in development [Flower89, Flower81, Meyer82].

4.2.1 Product plan

A product plan is, essentially, a working sketch of the document; it describes how the document will look. In it, the writer identifies high-level content and organization. The writer also designs the appropriate format for the audience and purpose. (In one sense, standards provide part of the product plan.) The product plan is the plan the writer makes for explaining content to the reader; it shows how the writer will communicate the information, including hierarchy and emphasis.

Writing experts have recommended many types of diagrams and charts to help writers translate their ideas into product plans, including traditional outlines and issue trees (or idea diagrams). Issue trees [Flower89, Olsen91] resemble Warnier-Orr diagrams. Since different forms work equally well, depending on the type of document and the writer's needs, writers should use the simplest form that will still handle the complexity of the information. For documents requiring extensive reviews during development or for documents written collaboratively, the type of plan may depend on what the reviewers or other writers can easily relate to.

4.2.2 Process plan

The process plan is the plan for the writer—the work plan or project plan. It contains procedures for completing the document and for implementing version control. In the plan, the writer determines how to achieve the goals, overcome the constraints, and resolve conflicts in the rhetorical situation (see Section 1). It also includes such information as schedules and a list of required reviews. For a short document, the process plan might not be written down. More formal plans become necessary when large, complex documents are involved and when writing is done collaboratively. Sometimes writers use project management tools, such as PERT and Gantt charts.

4.3 Generating text

Writers typically follow the nonlinear models described in Section 2.2. As they write, they continually check their work against their original product and process plans. They examine their problem definition and purpose statement. They further analyze the elements in the rhetorical situation; they reconsider the communication triangle and the interactions it reminds them of. These activities are not necessarily formally planned or explicit; they are cognitive activities that writers perform as they write.

Occasionally, a writer might work in a linear manner, following an ordered sequence of activities—analyze, design, generate, evaluate, revise, edit. The writer makes transitions from one phase to the next, working straight through the document using a detailed plan (whether or not it has been put into writing). Selzer provides an example of one engineer's linear composing process and notes that it is unusual [Selzer83].

Some writers, when they begin to generate text, experience a difficulty commonly known as writer's block. Rose notes how rigid rules and inflexible plans may prevent a writer from generating text [Rose80]. On many occasions, writers are unable to generate text because they become concerned with local issues, such as sentence structure, sentence order, or word choice, too early in document development. Strategies for avoiding writer's block include setting manageable subtasks and scheduling them to be accomplished in a realistic period of time. Writers also need to recognize that first drafts do not need to be well written or properly formatted and that it is appropriate to *satisfice*, that is, to accept a less than perfect expression and form, and get on with writing. Practical advice about getting started appears in [Olsen91, Tichy88, Flower89]. For more on the notion of satisficing, see [Flower89, Simon81].

The following strategies for generating text have been adapted from the software community (see [Levine91] for additional discussion). The choice of strategy may rest on individual preference, but it is also likely to be based on the writer's analysis of the particular writing task. The strategy can be matched to the goals and constraints of the writing project, and choice of strategy affects both the product and process plans.

4.3.1 Rapid prototyping

In software engineering, prototyping is a technique for providing a reduced functionality version of a software system early in its development. In writing, a prototype is a whole document with what can be called reduced functionality: it contains all the

essential elements of the target product, but it is not fully developed. The prototype is the skeleton of the document without all the elements fleshed out; it is developed without attention to the finer details.

Writers construct an early version of a document that users or peers can evaluate both informally and through formal review. Writers, thus, gain feedback early in the writing process, before investing too much time [Guillemette87, Taylor82]. The draft, or prototype, may be used once as a concept piece and thrown away or may be the basis for evolutionary development. In either case, prototyping allows writers to learn more about the requirements for the document and to better meet the needs of the readers.

4.3.2 Stepwise refinement and iterative enhancement

Stepwise refinement and iterative enhancement involve developing a document through ongoing drafts or iterations. Stepwise refinement is analogous to providing body text for a portion of an outline that was empty. It may also involve the creation of additional subsections. Iterative enhancement involves both adding text and improving the existing text. Each iteration is a successive approximation which comes closer to the final product.

This means the writer refines the document in steps, focusing on different goals at each step and moving from large issues to more local concerns as the document comes closer to its final version (*final* is defined by whatever criteria are appropriate for that particular document). Writers remain aware of choices they will make later while selecting certain issues to resolve in the present "pass" through the document. By making deliberate choices about what to attend to at each step, writers can solve big problems before investing time on smaller matters. (See Section 4.5 for further discussion.)

The above techniques allow for levels of completion. The writer has a complete version—of increasingly higher quality—at each step of the way. If organizational priorities shift or deadlines approach, the writer has the option of stopping at any time. The writer satisfices [Simon81, Flower89]: accepts text that is *good enough*, not because the writer prefers less to more but because there is no choice. Similarly, if the document will be used for a brief time by a few in-house people, the text need not be as polished as a document that will be widely distributed to

customers. Thus, writers can match the level of quality and amount of time invested to the document's audience and purpose and to the constraints of the rhetorical situation [Bitzer68, Hamlet86, Mingione83].

4.3.3 Risk-driven approach

Following the spiral model, some writers take a risk-driven approach to generating text, writing the hardest or most risky parts first. By doing so, they can identify the most difficult problems with the document. If the problems are too great, the project can be reevaluated and changed before significant time and money are invested.

Risk analysis is one aspect of task definition (see Section 4.1.2). The analysis can expose unrealistic goals and clarify constraints; and it plays a part in process planning, identifying possible scheduling and budget problems or potential problems with information gathering and verification.

Many writing texts advise writing the easy parts of the document first. This, they claim, will give the writer a sense of accomplishment early in the project. Although this is often good advice, it sometimes gives the writer a false sense of accomplishment and may cost the organization time and money when it is later discovered that the writing task was not well defined, resulting in the whole document being discarded and the task redefined.

4.3.4 Reuse

The notion of reusable components applies to writing as well as to software. Writers reuse parts of documents they have written before. However, wholesale reuse of text is appropriate only if the text suits the new context, audience, and purpose. More often, writers adapt portions of an existing document to suit the new situation [Selzer83].

Teaching Consideration: Students should be warned of the perils of a too liberal use of *cut* and *paste*. It is useful to give them experience in selecting text and adapting it to a new audience or purpose. They should also be aware of the rules involving reuse of other writers' text.

Another aspect of reuse is the development of documents with a modular structure containing units that can be reused. Examples are modules used in a multivolume set of documentation or those that can move from a requirements document to a specification or design document. Modules may consist of one sentence, a paragraph, or entire sections of text.

Introductory and background materials often are the portions that can be moved from one document to another. Reuse also relates to the development of templates, which provide the framework of a document without constraining the linguistic choices and without entirely limiting the content choices. Standards are one type of template.

4.4 Testing

4.4.1 Informal testing

Because evaluation is an integral part of writing, good writers test as they go. They check their work against their plan and evaluate, for example, whether the document addresses the problem and whether the problem has changed. Writers also repeatedly ask questions such as: Does the document meet the audience's needs? Is the organization appropriate for the audience? Is the text cohesive? Is the tone appropriate?

Writers also give unfinished versions to others for their response on specific aspects of the document. The test-as-you-go technique significantly reduces the time needed for formal testing. It also improves quality because drafts can be reviewed a number of times as they are refined, and the writer can select a diverse group of evaluators. An evaluator is likely to recognize gaps between chunks of information, lack of logical consistency, ambiguity that can result in misunderstanding, and areas of unneeded redundancy that writers, with their conceptual closeness to the text, often miss.

Evaluation is done on the *macro* level and *micro* level. Macro issues are global issues that address how the document works. They include the following: Is the problem/purpose clear? Is the document appropriate for its audience (content, level of detail, tone)? Are there gaps in logic? Is there missing or inaccurate information? Is the document explicit enough? Is the style easy to read? Is the level of formality appropriate? Do the organization and layout suit the way the document will be used?

Micro issues are local issues that have to do with fixing the product: Is the vocabulary appropriate? Is the word choice all right? Are any of the sentences or paragraphs unclear or awkward? Are there any errors in grammar or punctuation?

4.4.2 Formal testing

In addition to ongoing evaluation by the writer and others, more formal tests should be performed. For documents written for the public, writers perform formal tests on the alpha and beta versions of the document, if time and budget permit. Formal testing is also appropriate for critical internal documents.

Subjective tests include structured interviews (in which readers are asked focused questions), paraphrase tests, and protocols. Verbal protocols require the subject to think aloud while reading the document; motor protocols (user edits) require the subject to think aloud while performing a task described in the text. A formal technical review by subject matter experts is another form of subjective test.

Commonly used objective tests include readability formulas, style programs, and performance tests (speed and accuracy in performing specific tasks based on the reader's comprehension of the text). Duffy [Duffy85] and Selzer [paper in Anderson83], among others, caution against relying too heavily on readability formulas.

Because writers are so familiar with their subject matter, they risk making incorrect assumptions about the audience. It is difficult, for example, to determine what an inexpert audience knows about a technical subject. Therefore, tests performed by the end user of a document provide writers with particularly valuable information. These tests help writers to verify that the document meets the readers' needs, identify weaknesses in the document, and determine how to improve it [, Bond85, Bond85, Olsen91, Wright83].

4.5 Revising

Revision is an ongoing part of the writing process. Just as writers continue to analyze and evaluate as they write, they continue to revise. And they may revise their plans as well as their document. In some cases (for example, when requirements have changed or testing has uncovered serious problems) a revision may require a thorough redesign.

During revision, a document may go through a series of focused iterations (see Section 4.3.2). For example, in the early drafts, writers might concentrate on determining whether the information is complete and accurate, and whether the level of detail and level of difficulty are appropriate. They may look for

ways to combine pieces of information, or identify information that should be added or excluded. In addition to relying on their own judgment, writers use feedback from reviewers and formal test results as the basis for their decisions.

In later drafts, *levels of edit* become the focus. When using levels of edit, the writer makes multiple "passes" through a document for maximum efficiency. For example, one pass might focus on whether the tone is appropriate for the audience. During other passes, the writer addresses other high-level issues such as organization, logic, or cohesion. Later, the focus might be sentence structure or word choice, and so on, down to grammar and punctuation. Olsen provides several chapters on editing and one on proofreading [Olsen91]. Tichy gives editing advice and provides a section standards of correctness [Tichy88]. For a discussion of the differences between experienced and inexperienced writers, see [Sommers80].

4.6 Maintaining

Many documents produced in software organizations are *organic*, i.e., they are not truly finished when they are released but must change over time to reflect changes in the software or the operation of the system. Sometimes change pages are released to indicate local changes in the document; other times, the entire document is rewritten and re-released to reflect major changes, describe enhancements, or correct mistakes. If a document requires major redesign, maintenance might become a salvage operation to gather bits of usable text.

Technical writers are developing strategies for writing documents for ease of maintenance while keeping in mind problems of version control and configuration management. Reuse and modular construction are popular approaches. (See [Jones88] for an example.)

5 The Written Product

5.1 Principles of Linguistics and Discourse Analysis

This section is an introduction to the vocabulary needed to talk about the features of well-written documents and to explain problems in documents that are not well written.

Effective writers consciously or unconsciously apply principles of linguistics and discourse analysis when they write. Writers use these principles to generate, evaluate, and revise their text. For teachers, these principles provide a basis for evaluating student work and enable them to provide constructive feedback and assign grades.

Principles of linguistics and discourse analysis differ from traditional handbook rules of usage, grammar, and punctuation. Handbook rules tell a writer how to avoid being wrong according to the current conventions. (Williams has a useful and realistic discussion of rules, nonrules, optional rules, and *betes noires* [Williams89].) In contrast, principles of linguistics and discourse analysis help writers make choices that enable them to communicate effectively with their readers. These principles are based on research on how readers process information. They are applicable even when the writer must follow a template or use a formal standard (see Section 1.4.2.1).

General references for Section 5.1.1 and Section 5.1.2 are [Olsen91, Brown83], which contain nearly all the information an instructor is likely to need. Additional sources, when necessary, are cited within individual subsections.

5.1.1 Global concerns

Cohesion is the way elements of the text (units of thought, sentences, paragraphs) are tied together. These explicit ties, which are part of the surface structure of the text (see Section 5.1.2) help readers to understand the connections between ideas and between parts of the document. Some ways of achieving cohesion include hierarchical structure, parallelism, and word choice.

Coherence, which moves beyond issues of cohesion, involves the underlying semantic unity of the text. Coherence, in a sense, deals with the ties between the text and the reader. In Lessons 2 and 3, Williams describes how writers can effectively convey meaning; that is, how they can make their sentences work together throughout the whole document by revealing to the reader bit by bit, in a logical sequence, the meaning that the writer would like to convey. Williams' recommendations, which include using tightly linked sentences and paragraphs, explain how writers can increase the likelihood of readers understanding and decrease their chances of misunderstanding [Williams89].

Hierarchy is important for highlighting information. Christensen provides concrete information about effective sentence and paragraph structure [Christensen78]. When material is arranged in a hierarchy that is clear to the reader, readers' comprehension, as well as their speed of comprehension, is greater. One reason is that top-level content is more prominent and receives more attention from the reader [Meyer82]. Readers call top-level information to mind frequently as they tie in details.

Writers identify hierarchy when they are planning a document. As they write, test, and revise, the ways they communicate the hierarchy may change and sometimes the hierarchy itself may need to change. Outlines and issue trees are tools for both planning and evaluating hierarchy. Writers can derive (or have test subjects derive) an outline or issue tree from a text to determine its completeness and the effectiveness of its arrangement [Flower89, Olsen91]. See Section 4.2.1 for a discussion of product plans, including outlines and issue trees.

Although there are other ways to highlight information, such as lists and typography, they have not been specifically addressed in this module. For more information, see Huckin's paper in [Anderson83] and the references listed in Section 6.2.

5.1.2 Local concerns

Writing sentences. Linguists refer to the meaning of a sentence as its deep structure and call the grammar of a sentence its surface structure. The deep structure is the idea in the writer's mind, and the surface structure is the written sentence. Writers transform deep structures into various surface forms, choosing one that effectively communicates their meaning to their readers. For information on effective sentence development, see [Tichy88, Williams89, Andrews82a].

Functional sentence perspective (FSP) is a way to approach sentences in terms of their larger context—it provides heuristics for revising on the sentence level while keeping the more global issues in mind. It is another source of cohesion (see Section 5.1.1). Good coverage of functional sentence perspective can be found in [vande Kopple82, Olsen91]. vande Kopple provides guidelines for revision, and Olsen offers many examples and some exercises. Williams embeds similar information in several of his lessons [Williams89].

Grammar and punctuation. Issues of punctuation and grammar are covered in many texts and handbooks [Tichy88, Williams89, Olsen91, Chicago82]. Good writing should be correct. Error undermines the writer's credibility, and some readers simply will not read text that is riddled with error. However, a document can be error-free and still not achieve its purpose. Tichy points out, "Avoidance of error does not of itself constitute good writing."

When consulting handbooks, writers will find that the rules of usage change over time, and, more confusing, authorities do not always agree on what the rules are at any given time. Williams gives advice on dealing with this problem [Williams89].

Spelling. Spelling is an issue that can be handled in part by spell checkers on word processors. However, writers should understand the limitations of the technology; for example, a word misspelled to be another word will not be caught (it/its; garage/garbage). Use of a spell checker does not eliminate the need to proofread.

6 Other Considerations

The following sections note two areas of concern that are highly relevant to technical writing but move beyond the fundamentals—collaborative writing and document design. These sections are a starting point for instructors who wish to extend their writing instruction to address these topics.

6.1 Collaborative Writing

Collaborative writing is becoming increasingly common, especially in projects that involve large teams; and the subject is receiving new attention in writing research. Collaboration takes a number of forms; the following are examples:

- Text is generated in "chunks" by individuals, then spliced together later, with one person designated as editor.
- Two or three people work on a single piece of text; one also acts as a recorder of ideas. They produce a prototype document; each contributor reads and comments; and each edits the later drafts, performing the level of edit appropriate to his or her strengths—from the technical expert who verifies content to the language expert who checks grammar and punctuation.
- A software practitioner or group works with a technical writer,

each contributing expertise during all the activities of the writing process described in Section 4 [Dunkle88].

For more on collaboration, see [TechComm91, Houp92, Bruffee84, DohenyFarina86, Kraemer88].

6.2 Document Design

Document design is an "umbrella term" that covers the verbal and visual aspects of written documents and the ways they work together.

Document design issues can be divided into three categories, though there is particular overlap between the last two. The following list contains a sample of the topics in each category and pointers to appropriate references.

1. *Visual aspects of written language*: Topics include typography; appearance on the page (margins, columns, length); and physical arrangement of verbal cues (headings, lists, etc.).
   [Houp92, Felker81, Olsen91, Rehe81, Redish89, Brusaw76, Duffy81]
2. *Graphic representation of information*: In technical writing, the most common are graphs, tables, and diagrams.
   [Tufte83, Olsen91, Brusaw76]
3. *Combining words and graphics*: Topics include graphic cues (bars, symbols, etc.), page layout and formatting; deciding when to use words and when to use graphics; understanding how written text and graphics work together, and how people read and use them.
   [Tufte83, Watzman87, Redish87]

Several authors discuss software issues specifically [Bentley86a, Bentley86b, Baecker88].

Glossary

discourse
verbal expression in speech or writing.

discourse analysis
the study of expression in speech or writing. Discourse analysis focuses on language in use, not formal properties of language independent of purpose or function. The term often covers research in intersecting disciplines, including sociolinguistics and psycholinguistics (see *linguistics*).

document design
a term that refers to both the verbal and visual aspects of a document and the ways they work together. Document designers are concerned with creating documents that communicate clearly and are easy to use. Guidelines for document design, which are primarily based on principles of cognitive psychology, take into consideration factors such as readers' comprehension and the usability of the document.

heuristic
a guideline, not a rule, used in solving a problem or carrying out a process.

lexical cohesion
meaningful relationships between words and/or smaller word units (morphemes); these relationships connect sentences, paragraphs, and larger units of text in ways that tie the text together.

linguistics
the study of the nature and structure of language. Psycholinguistics is primarily concerned with language comprehension. Sociolinguistics places stress on social context and the social interactions that are a part of language use.

positivism
the theory that knowledge is based on natural phenomena as verified by empirical science.

protocol
the results of a research procedure in which individuals are tape recorded as they think aloud while they are performing a reading or writing task.

rhetoric
the study of the elements (including invention, arrangement, and style) used in writing and speaking; effective expression, created by a communicator who has taken context (including purpose and the

characteristics of the audience) into consideration; the persuasive use of language.

satisfice

in general, to accept an adequate but less than perfect solution to a problem; in writing, to accept a less than perfect expression or form and continue with the writing process; to determine that a solution or expression is good enough for a given situation.

writing across the curriculum

educational position that supports the teaching of writing in courses offered by many academic departments; the integration of writing instruction into content courses on many topics, not only composition courses.

Teaching Considerations

Using the Annotated Outline for Teaching Writing

This module outlines a body of knowledge about writing. Although the order of topics gives the instructor an understanding of the field, it isn't the optimum order for teaching. The list below identifies the intended audience of each section of the annotated outline and gives a brief explanation of its content. Sections 1, 4, and 5 contain the material that will help students learn to write effectively; Sections 2 and 3 are primarily background for the instructor. Grading criteria can be drawn from Sections 4 and 5.

Section 1. Rhetorical Situation – teachers and students

This section discusses the larger contexts within which the writing process takes place. Writers who don't take context into consideration are unlikely to be effective communicators.

Section 2. Views of Writing – teachers

This material, which gives a broad picture of the field, is background for teachers. It has two purposes: to help them identify how our material fits into the field in general and to provide a framework for understanding the relationships among the references we cite.

Section 3. Analogies – teachers (primary), students (secondary)

The section is, essentially, a bibliographic essay—an introduction to what has been written on the similarities between software development and writing. We present the information as a starting place; we intend for teachers to extend these analogies and, most likely, develop new ones. The analogies provide a perspective on the writing process that will allow teachers to tailor instruction to software engineering students.

Teachers should encourage students to explore the analogies whether or not they assign readings from the section. By making connections between their technical field and the field of writing, students will learn more efficiently, perhaps in both domains. They will more easily transfer software skills to writing tasks, and they are more likely to view writing as relevant part of their professional development.

Section 4. The Writing Process – teachers and students

While the analogies section provides the stimulus for learning, this section provides the substance. Because the material is about *doing* rather than *knowing* a hands-on, workshop approach is essential. Students will benefit most from short lectures, much practice, and much feedback.

Section 5. The Written Product – teachers and students

Readers often know instinctively whether a document is good or bad, effective or ineffective. The material introduced in this section enables a reader, especially the teacher, to diagnose writing problems and find solutions. The section introduces functional principles and a vocabulary for discussing why a document isn't good and explaining how it can be better.

Objectives

By the time students complete their program of study, they should have a firm understanding of the material in Sections 1 and 5 and should have mastered most of the material in Section 4. (One exception is the material in Section 4.4 *Testing*— students need to be aware of the need for testing and know that formal tests exist, but few instructors will have the time to teach students how to perform these tests.)

Instruction based on the material in this module should address the objectives listed below. It is difficult to set priorities on these objectives. In teaching writing, it's not so much a matter concentrating on some objectives and leaving others to be achieved in other courses; all the objectives must be addressed at some level. Students who gain only content knowledge will not necessarily have learned how to write effectively. Rather, as they learn more about writing—and receive feedback on the writing they do—students will achieve each objective to a greater degree.

Affective

All students should achieve the following objectives after receiving instruction based on the material in this module:

- Realize that writing and software development are problem-solving activities and that skills used during software development resemble those used in writing.
- Appreciate the importance of writing in the software development process and, especially, understand the problems that result from seeing documentation as an add-on function rather than an integral part of the software life cycle.
- Appreciate the importance within the writing process of analysis, planning, testing, and revision.
- Realize that technical communication is rhetorical (i.e., that writing is done in a context that affects writers' choices) and that simply following a formula will not guarantee a good document.
- Become aware of the difficulty, complexity, and effort involved in writing a precise and readable document.

Cognitive: Knowledge/Comprehension/Application

The objectives in this section represent the foundation for achieving the goals listed in the next section, *Analysis/Synthesis/Evaluation*. Students will be able to:

- Define or describe components of the rhetorical situation and the communication triangle.
- Identify the components of their rhetorical (writing) situation: Who is the primary audience? Who are the other audiences? What are the constraints on the document, writer, audience? What are the goals for the document, writer, audience?
- Apply their understanding of discourse communities to identify some of the characteristics of the software engineering discourse community.

Cognitive: Analysis/Synthesis/Evaluation

Students who achieve the following objectives are likely to become effective writers in the workplace. They will be able to do the following:

- Analyze the rhetorical situation before writing a document; evaluate the appropriateness of a draft for an identified rhetorical situation.

- Plan a document and revise that plan when necessary.
- Create an architecture for a document (that is, an overall design) and revise that structure when necessary.
- Write and revise a document, beginning with a problem definition.
- Use linguistic principles to analyze and evaluate a draft.
- Participate effectively in a peer review.

Metacognitive

The following goals have been labeled *metacognitive* because they identify the ways in which students should be able to think about their own writing process.

Students will gain self-awareness about their own writing process; they will be able to:

- Identify the planning techniques they use and informally evaluate the efficiency of those techniques.
- Identify the organizational (architectural) techniques they are using and informally evaluate their effectiveness.
- Identify their strategies for structuring a document and generating text, and informally evaluate the effectiveness of those strategies.
- Explain their rationale for revision.

Prerequisite Knowledge

For teachers: The module and the references it cites provide sufficient information for a software engineering instructor to incorporate writing into a technical course. Although no prerequisite knowledge is required, teachers will need varying amounts of time to prepare to teach writing. Familiarity with areas such as rhetoric, linguistics, and cognitive psychology will decrease the amount of preparation that will be necessary. It is helpful—and recommended—but not essential to seek advice from experienced teachers of writing.

For students: No prerequisites are required, but we assume that students have some familiarity with software development. A previous writing course is helpful, especially a technical writing course. Courses that focus on the process of writing would be most helpful.

Primary Texts

If students are to use only one book, we suggest [Olsen91]. It contains material on a substantial number of the topics addressed in this module. See the bibliography for further information about this text.

If students are to purchase more than one book, we recommend [Tichy88] and [Williams89]. Tichy is down-to-earth and thorough, but she doesn't provide practice exercises. Williams is equally sensible and a bit more sophisticated; his book contains exercises but is not as complete. All three books could be used throughout an academic degree program and as references on the job. Inexperienced writers may find [Flower89] helpful, but they may be offended by the simplistic language and examples that are geared to freshmen.

Supplementary Material

In addition to the recommended textbooks, we suggest that the following books be put on reserve in the library or excerpts distributed to students: [Felker81, Mattes76, Tufte83, Anderson83, Odell85]. These selections contain good supplementary material for both students and teachers. [Felker81] provides guidelines for writing and document design, along with a brief description of the research on which each guideline is based. The strength of [Mattes76] is in the way it ties problem statements and audience analysis to organizational settings. We highly recommend [Tufte83] for its information on how to present scientific and technical data in charts, tables, and graphs. [Anderson83] and [Odell85] are considered classics in the technical writing field. They include essays and reports on research.

Another source of material is the Society for Technical Communication (STC). The proceedings of their annual International Technical Communication Conference (ITCC) contains a wealth of information on subjects such as technology and visual arts as well as writing, editing, and teaching. Although the quality of the contributions varies, there is more consistency among the academic sources. The STC also publishes a journal, *Technical Communication*.

Members of ACM SIGCSE (the Association for Computing Machinery - Special Interest Group on Computer Science Education) occasionally present their experiences with teaching writing in technical courses. The group holds an annual conference and publishes *SIGCSE Bulletin*.

Other sources include the following journals: *Journal of Technical Writing and Communication, Technical Communication Quarterly* (formerly *The Technical Writing Teacher*), *Journal of Business and Technical Communication,* and *College Composition and Communication*.

Incorporating Writing into the Software Engineering Curriculum

We believe that students should learn to write by working on the same types of documents they will write as software professionals. In software engineering courses that already include these documents, instructors do not need to assign special papers. Rather, instruction will be most effective when the material in this module is tied to the content, writing activities, and assignments that are ordinarily part of the technical courses. For example, if students are asked to write a requirements document or a project plan, that document should be the basis for the writing instruction. (For a discussion of skill transfer, see the *Philosophy* section in the preface of this module.) Criteria for grading students' work should address both the technical content and the quality of the writing.

We advise against teaching writing in a separate course. If students do receive instruction in a separate course, the material must be reinforced in the students' technical courses. Only this kind of continued effort will address the problems of ineffective communication.

The following sections contain suggestions on how to incorporate writing into software engineering courses and how to evaluate student work. We offer these suggestions as a place to begin. Additional advice and examples from our experience appear in [Pesante91, Levine91]. For examples of other teachers' activities, see [Hartman89, Rice84, Meinke87].

We encourage you to exploit your knowledge of software development in order to place the material in this module into context for your students and to develop additional analogies and examples.

Providing instruction

Ongoing attention to writing issues is more effective than direct instruction with little or no follow-up. Here is one example of what we mean by ongoing attention: In every course students should be introduced to or reminded of the communication triangle (see Section 1 of the annotated outline). Since the communication triangle does not explicitly address purpose, students should also be reminded that every document has a purpose and that someone will read it. Even class notes are read by the person who wrote them—and many students will have had the experience of rereading and not understanding their own notes. This introduction may take five or ten minutes of one class period. Then, when students must write a document, they should be asked to identify the reader, subject matter, and purpose of the document. You may ask them to write this down, or you could ask the question and get their answers when you explain the assignment.

If you ask students to provide written information about themselves on the first day of class, you could begin your instruction there. The document involves a writer (the student), a reader (you), subject matter (the information you request), and a signal (written English language). It's up to you to explain the purpose since you have made the request. Teaching by this example doesn't take much class time, but it is very effective. Students who are consistently exposed to the same kind of instruction and concerns will gradually gain an understanding of rhetorical situations and skill in applying that understanding when they write.

Defining the term *document*

Keep in mind that *document* is a very broad term. All of the following are documents. They are written for readers and to achieve a purpose.

- The information sheet you request on the first day of class, even if you merely ask for name and user-id on an index card.
- A bulleted list identifying the requirements for a software system.
- The minutes of a planning meeting.
- The written results of a technical review.
- A summary of a journal article.
- A progress report.

When students become aware that everything they write is a document, they have taken the first step beyond thinking about writing only in terms of formal papers or graded work.

Instructors should be aware, too, that every piece of writing they give to their class is a document and that the standards they set (or do not set) are primary examples for the students. Instructors must demonstrate that they mean what they say about quality writing. Good quality is essential for documents such as:

- Course outlines.
- Course descriptions in the catalogue.
- Homework assignment statements.
- Long-term project requirements.
- Software/document requirements.
- Administrative memos.

Selecting topics

It is not possible for instructors to address every writing issue in every assignment. An exception might be a document written over a longer period of time; for example, a user's guide that is assigned early in a course and is due at the end of the course. Regardless, it is more feasible to ask students to pay attention to different aspects of writing at different points in the course.

Students must write drafts of their documents, so the same document can be used to address a number of writing issues. For example, students might concentrate on purpose statement and organization in one draft of a document, appropriateness for audience and purpose in another, and style in another. The goal should be to bring more and more skills to bear on the writing process.

Similarly, it is not likely—or necessary—that instructors give the same amount of attention to each writing topic in all courses. For example, a teacher might stress task definition and audience analysis in a course on requirements or specifications and only mention techniques for generating text.

More examples

Other examples of incorporating writing instruction into regular assignments:

- Require a written product plan for one document students normally write for the course, or develop the plan in class before students begin writing.
- Discuss strategies for generating text. This can be done in one block of time or by introducing one strategy at each of several class meetings. Ask students to identify which strategies they have used, effectively or ineffectively, either in your class or in another.
- Provide several examples of a document students will write and ask them which ones are good, and why. You can address both content and writing issues this way. Following [Knuth88], provide examples of programs—they're documents, too—and do the same thing.

Specific Teaching Techniques

It can't be denied that teaching writing involves substantially more work than not teaching it. To become effective writers, students need to write drafts and get feedback on those drafts; they need to revise their work based on analysis and evaluation. They also need to receive credit for these activities, not simply a grade on the final product. Fortunately, many of the techniques for teaching writing effectively also help control the workload.

Team teaching

Team teach with a member of the English department who is familiar with the material in this module; that is, one who can draw on the analogies between the domains and provide process-based instruction. Knuth, among others cited in the bibliography, taught a course with a technical writing instructor; he describes in detail the logistics of that arrangement and the roles each instructor played [Knuth88]. Team teaching has also been used in Carnegie Mellon's Master of Software Engineering program—a computer science professor and one of the module authors recently co-taught a software development seminar.

In a modified version of team teaching, arrange for an English teacher to provide feedback on drafts and grade the final product. The technique is particularly effective if this teacher speaks to the class about writing issues related to the feedback they receive.

Another technique is to seek advice from a technical writer or a teacher of technical writing. At Carnegie Mellon, an instructor worked with one author of this module to structure writing assignments, determine the focus of each, and select ways of providing feedback. A few hours of consultation (in this case, less than six hours for the semester) can be a great help. See [Pesante91] for suggestions on getting help with writing instruction.

Demonstrations

Through demonstrations, you show your students how to be effective communicators. Demonstrations can take many forms, such as the following:

When you give a writing assignment, start it for the students either with a handout or on the board. It's not unusual for a writing assignment to provide a problem definition or identify the audience and some of the constraints. You might choose to use a portion of the assignment for an in-class exercise that must be completed later. Select an item from the analysis or planning sections of this module (Sections 4.1 and 4.2) as the focus for the exercise. Or, later, select a paragraph or

two from students' drafts and revise using an overhead projector and transparencies. For pedagogical and other reasons, choose examples that contain a typical problem, not one that is peculiar to an individual student.

Demonstrate the writing process by creating and working with a short piece of text (about 100 words). In 20 or 30 minutes, the class should be able to produce a draft and evaluate and revise it several times.

Bring examples of your best writing to class and discuss what you have done. In his mathematical writing class, Knuth regularly discussed his writing with his students [Knuth88]. We have used this technique in our own teaching: each week, students received a new draft of the instructors' work, along with a five-minute description of how the paper had changed and why. Before long, the students could identify many intervening activities: refining the task definition, learning more about the audience, generating new ideas, changing the schedule, responding to reviewers' comments, overcoming writer's block, etc.

Provide outside examples of how people write and how they read. Flower provides two small samples [Flower89]. Faculty in the psychology department, English department, or education program of your school might also be able to provide samples or information about other sources. Also consider making your own "think-aloud" tape the next time you plan, write, or revise a paper; it is sure to hold your students' attention. If you don't care to tape your own process, perhaps some of your colleagues or members of your writing faculty would be willing. Tapes of collaborators are particularly interesting.

Student involvement

Self-evaluation precedes peer review, but students can critique and teach each other. Working in pairs or in small groups, they can provide feedback on every activity in the writing process (peer conferencing) and evaluate each other's drafts (peer review). Students who are new to these activities may need clear structure—a peer review form with specific questions, for example. A few students may need to be reminded to be sensitive to the writer's feelings when offering feedback; and it should be clear to all students that the purpose of these activities is constructive—to provide information that will help them improve their documents.

Students can also gain valuable feedback when another student "reverse engineers" a draft, for example by creating an outline or issue tree from the document (Section 4.2.1 discusses issue trees). Problems with organization and hierarchy often come to light through this technique. For some documents, it may be appropriate to ask students to have a member of their intended audience do a think-aloud protocol or perform a task on the basis of their written instructions.

Other activities:

- Ask students to write a problem or purpose statement for the paper they are reviewing or to infer the characteristics of the intended reader.
- Have students do a series of quick reviews of the same draft. They might read once looking for extraneous information and gaps in information, once for ambiguity, once for sentences that are awkward or that violate linguistic principles (see Section 5 of the annotated outline), and once for errors in grammar and punctuation.
- When an assignment is complete, students can benefit from one another's experience by receiving feedback *en masse*. After you have graded the final document, duplicate examples of typical problems (with students' names removed) and ask the class to diagnose problems and suggest improvements. It's also instructive to provide samples of effective writing; ask the students to explain why particular samples are effective—and ask them to suggest further improvements.

Practice

Practice is valuable even when teachers don't grade, or even read, every piece of writing. In addition to relying in part on peer reviews, you can have students keep portfolios of their writing and use spot grading (in the sense of spot checking) to encourage practice. See [McLeod88] for a description of this technique.

Evaluating Writing

Writing should account for a percentage of the grade on every technical document that students write, and it should account for a percentage of the final course grade. Moreover, instructors should take into account their students' activities during the writing process (see Section 4) as well as grading the written products (see Section 5).

Evaluating the process

Students should be required to submit drafts and notes from peer reviews or user tests when they submit the final document. Since students need to receive feedback throughout the writing process, not just at the end, they should be rewarded for using this feedback to improve their work. They should also be rewarded for providing constructive feedback to others, and for being able to evaluate their own work and make improvements.

Other factors in the grade for writing might include: credit for providing peer reviews for other students, conducting tests, and participating in classroom activities. This credit can take the form of checkmarks or points rather than letter grades. The chart below illustrates two systems used by the authors of this module. We convert the accumulated checks or points to a percentage of the final grade.

Quality	Checkmarks	Points
exceptional	✓✚	3
acceptable	✓	2
poor (or incomplete)	✓–	1
not done	no credit	-1

Evaluating the product

Evaluating writing is not a matter of determining right and wrong so much as it is a matter of noting what works—and what works best out of a number of options. Often, evaluators know instinctively that one paper is clearer and easier to read than another. Linguistic principles provide good criteria for identifying why a document "works." The areas of document design, discourse analysis, and rhetoric also provide criteria for giving feedback on drafts and awarding a final grade on the finished document. Although students should be evaluated primarily on the effectiveness of their writing, they should also be graded on correctness—spelling, punctuation, and grammar. (See Sections 1, 4, 5, and 6.2 of the annotated outline.)

Another consideration in determining a grade is whether the final product is better than the earlier drafts. Two-stage grading is one way to recognize the values of drafts and final versions. Both are graded, but the final effort is worth twice as much as the draft. The grade for the draft recognizes and encourages a good start; the grade for the final version recognizes and encourages improvement.

Below are two approaches to grading documents: *wholistic grading* and *detail grading*. Each has a place in evaluation, and both are valid.

In wholistic grading, the instructor reads each paper quickly and assigns a grade without going back to diagnose specific problems. The wholistic grader usually writes a comment on the end of the paper. Wholistic grading saves time and allows the teacher to respond to overall quality and global issues such as logic, organization, coherence, and tone. However, low-level errors such as grammar and spelling errors may be overlooked; and only extremely awkward sentences may be noticed.

In detail grading, the instructor reads each paper carefully, identifying strengths and weaknesses along the way. The detail grader usually writes many comments in the margins of the paper and circles (or corrects) problems and errors. Sometimes a marginal notation can be tagged to a fuller comment at the end of a paper, and sometimes students are referred to a specific section in the textbook. Detail grading, though it takes more time, provides the student with concrete feedback. However, this approach often leads instructors to focus on local issues and neglect global ones.

A compromise would be to assign a wholistic grade to a document and read it a second time for a particular set of details. Other approaches are to use wholistic grading at one time and detail grading at another, or to give two grades on a single assignment.

Section 5.1.1 of the annotated outline provides information about global issues, and Section 5.1.2 addresses local issues. The technical writing textbooks we have recommended are good sources of information about both.

Exercises and Support Materials

Documents that students typically write for software engineering courses should be the basis for writing instruction. The following are presented for instructors who wish to include supplementary exercises and material.

[Olsen91]

Chapter 1, “Why Study Technical Communication,” contains excellent support material for motivating students who don’t believe engineers need to be good writers.

Part V, “Readability,” is a good source of reading assignments and exercises. Software engineering students will also benefit from the exercises in Chapter 17, “Instructions, Procedures, and Computer Documentation.”

[Tichy88]

Tichy's advice and examples can be selected as supplementary readings when specific topics arise. We particularly recommend Part 3, “Style,” and Appendix A, “Fallacies to Forget.” The book also addresses levels of edit (Ch. 2), outlining (Ch. 4), and standards of correctness (Ch. 6-9).

[Williams89]

Williams provides a good introduction to some basic principles. However, most of the exercises focus on local issues. Exercises can be selected based on the problems that are evident in students' writing.

[Flower89]

The self-exam (pp. 46-48) and the checklists for structure and diagnosing problems (pp. 131 and 257) provide the basis for several exercises.

Examples of think-aloud protocols appear in Chapter 2.

Issue trees can be found in Chapters 1 and 7.

Other material that students might find helpful are the descriptive models of the writing process (p. 52) and information about cues for the reader (p. 256).

[Pesante92]

The SEI Technology Series contains a videotape that introduces many of the concepts presented in this curriculum module. Because it features statements by software engineering students, the tape can be used as motivational tool as well as an introduction to writing.

Bibliography

Anderson83

Anderson, Paul V., Brockman, R. John, and Miller, Carolyn R., eds. *New Essays in Technical and Scientific Communication: Research, Theory, Practice.* Farmington, N. Y.: Baywood Publishing Co., 1983. Baywood Technical Communication Series, Vol. 2.

The editors rightly maintain that these twelve essays represent the best of current scholarship. The volume is divided into five parts: (1) Empirical Research, (2) Reassessing Readability, (3) Approaches from Rhetoric, Discourse Theory, and Sociology, (4) Historical Perspectives, and (5) Redefinition. Parts 2 and 3 will be the most useful background reading for the instructor, especially the following: Flower, Hayes, and Swarts' "Revising Functional Documents: The Scenario Principle," Huckin's "A Cognitive Approach to Readability," Winkler's "The Role of Models in Technical and Scientific Writing," Zappen's "A Rhetoric for Research in Sciences and Technologies," and Selzer's "What Constitutes a 'Readable' Technical Style?" "What's Technical About Technical Writing?" is a fine conclusion that raises thoughtful distinctions between technical writing and writing technically.

Andrews82a

Andrews, Deborah C., and Blickle, Margaret D. *Technical Writing: Principles and Forms*. New York: Macmillan Publishing Co., Inc., 1982.

This was one of the first textbooks to take a process approach (define, describe, and so on). It has good examples but is more appropriate for younger students.

Andrews82b

Andrews, William D., and Andrews, Deborah C. *Write for Results*. Boston, Mass.: Little, Brown, 1982.

The authors provide a number of charts and graphs to explain aspects of the writing process. They raise questions about what audience analysis really means and consider the implications and applications of such an analysis. The book also presents the connection between audience and purpose through an example of the same subject matter written for five different sets of readers.

Atlas81

Atlas, Marshall A. "The User Edit: Making Manuals Easier to Use." *IEEE Transactions on Professional Communication PC24,* 1 (Mar. 1981), 28-29.

Abstract: Possibly the simplest way to make a technical manual easier to use is a "user edit"—that is, having an inexperienced user try to work with a machine, using only its manual as a guide. His errors and hesitations should tell you where the weak points are. This report describes how to set up such tests, what to be careful of, and some of the benefits you can expect.

A short, succinct, and useful article on *informal and formal* testing with the user edit. Students writing technical manuals should understand that this testing is necessary, and not ideal, practice.

Baecker88

Baecker, Ronald. "Enhancing Program Readability and Comprehensibility with Tools for Program Visualization." *Proceedings of the 10th International Conference on Software Engineering,* 1988, 356-366.

Abstract: In order to make computer programs more comprehensible, the presentation of program source text, program documentation, and program execution needs to be enhanced over its conventional treatment. The paper describes a number of new techniques and tools developed to achieve these ends. One of these is a novel design for the effective presentation of source text in the C programming language using high-quality digital typography, and a processor which implements the design. Some experimental evidence is summarized to demonstrate that the resulting source text presentation is significantly more readable and comprehensible than the presentation conventionally used today. Brief descriptions are also given of two other techniques, the development of a novel system of structured program documentation incorporating both texts and graphics, and the portrayal of program execution with coloured computer animation.

Baecker's hypothesis "that a program's appearance dramatically affects its comprehensibility and usability" is empirically confirmed; subjects' performance, as measured on a comprehension test, increased by 25%. There is brief but good coverage of the design principles that have guided the experimentation and the recommended framework that applies these principles. Baecker also touches on the issue of programs as publications, noting that this work "represents another step toward Knuth's goal of literate programming."

Barker88

Barker, Thomas T. "Feedback in Hightech Writing." *Journal of Technical Writing and Communication 18,* 1 (1988), 35-54.

Abstract: This article is concerned with reviews, surveys, tests, and other formal procedures used in writing for the computer industry that are designed to provide authors and publications managers with information about the quality and nature of documentation. The literature in this area reveals a number of problems with feedback in hightech writing, including the lack of a consistent definition of feedback processes. The article investigates various types of reviews, theoretical aspects of feedback, and elements of feedback specific to hightech writing. This investigation yields three consistent perspectives on feedback: management, style and rhetoric, and research.

This article treats feedback practically by looking at current methods and theoretically through discussions of communication theory and automation. Barker's effort is mostly definition and survey; there's no "how to" here. But the article may be useful for comparisons between technical reviews and testing documents; "another of the parallels between hightech writing and software development is in the teamwork characteristic of both."

Barnum84

Barnum, Carol, and Fischer, Robert. "Engineering Technologists as Writers: Results of a Survey." *Technical Communication 31*, 2 (1984), 9-11.

Abstract: This article presents the results of a 1982 survey to learn the importance of communication skills to engineering technologists on their jobs. Similar surveys have assessed the importance of communication skills for engineers and for technicians, but none has polled engineering technologists. Knowing the attitude of students—"Why do we have to take so many English courses?"—the authors wanted to know whether that attitude changes after these students begin working. It does, they say.

The authors report on the 20% return from a sample of 1500 Southern Tech graduates representing "technology degrees in the civil, electrical, mechanical, industrial, architectural, textile, and apparel fields." These results should alert students to the importance of good communication skills; for example: 91% felt that their writing was either "important" or "very important" to their work; 73% noted that advancement involved an increase in their own time spent writing; and 75% rated organization of ideas as the "most important" skill needed on the job. Additional survey results on technical communication can be found in [Olsen83].

Bentley86a

Bentley, Jon, and Knuth, Don. "Programming Pearls." *Communication of the ACM 29,* 5 (May 1986), 364-369.

The value of this article lies in the concept of literate programming. Philosophically, literate programming emphasizes the place of literacy in software development, making room for the intersection of natural and computer languages. By highlighting the need for individuals to "read" programs, Knuth introduces a new kind of learning in software engineering education—one that stresses the importance of sharing knowledge by publishing (model) programs for emulation and enhancement. Finally, literate programming draws attention to the issue of audience so that, as Bentley observes, Knuth's work takes an "important step towards programs fit for man and computing beast." For an example of a literate program by Knuth, see the next "Programming Pearls" column (June 1986). For an additional reference to literate programming, see [Knuth84].

Bentley86b

Bentley, Jon. "Programming Pearls." *Communications of the ACM* 29, 9 (Sept. 1986), 832-839.

Bentley shares a few document design lessons on tables, figures, and text. He notes that iteration, consistency, and minimalism are "fundamental principles for producing better text, programs, or documents." A "catalog of pet peeves" is included in this brief, friendly introduction.

Bitzer68

Bitzer, Lloyd. "The Rhetorical Situation." *Philosophy and Rhetoric 1*, 1 (1968), 1-14.

Bitzer identifies three components that are essential to the rhetorical situation: the speaker or writer's exigence (or sense of an imperfection that needs to be addressed), the audience, and the constraints. Constraints include "artistic proofs," aspects of writing that the writer manages, and "inartistic proofs" such as contracts, agreements, laws, etc. This theoretical article is dense at times but valuable in providing a conceptual framework for understanding communication and rhetorical discourse. For instructors only.

Bizzell82

Bizzell, Patricia. "Cognition, Convention, and Certainty: *What We Need To Know About Writing*." *Pre/Text* 3, 3 (1982), 213-243.

Bizzell discusses discourse communities in the context of critiquing innerdirected (cognitive) theories of the composing process. She criticizes the process theorists (represented by Flower and Hayes [Flower81]) concerning their limited preoccupation with how people write and not why they write as they do. Bizzell argues that we must see writers as problem solvers "situated in discourse communities that guide problem definition and the range of alternative solutions." The final one third of her article is fairly repetitive in its discussion of the politics of the composition classroom. Supplementary reading for the instructor.

Bond85

Bond, Sandra J. "Protocol-Aided Revision: A Tool for Making Documents Usable." *Proceedings of the 1985 IBM Academic Information Systems University AEP Conference*, June 1985, 327-334.

Abstract: Participants will learn how to use Protocol-Aided Revision (PAR) to analyze and improve documents for clarity and usability.

This paper was the basis for a workshop session on protocol-aided revision. Topics include: "What is a protocol?" and "Why test?" Bond provides step by step instructions for conducting a protocol, analyzing the results, and applying those results to revision.

Brackett90

Brackett, John W. *Software Requirements*. Curriculum Module SEI-CM-19; DTIC: ADA235642. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., January 1990.

Capsule Description: This curriculum module is concerned with the definition of software requirements—the software engineering process of determining what is to be produced—and the products generated in that definition. The process involves all of the following: (1) requirements identification (2) requirements analysis (3) requirements representation (4) requirements communication (5) development of acceptance criteria and procedures. The outcome of requirements definition is a precursor of software design.

Brackett clearly demonstrates the role of communication in requirements engineering. He discusses the importance of understanding the needs of the software developers and the users of the software product.

Brown83

Brown, Gillian, and Yule, George. *Discourse Analysis*. Cambridge: Cambridge University Press, 1983.

This text provides information about linguistics and discourse analysis that is useful for instructors but probably too advanced for students. The topics of particular interest include: paragraphs, topic, information structure, cohesion, and coherence.

Bruffee84

Bruffee, Kenneth A. "Collaborative Learning and the 'Conversation of Mankind.'" *College English 46*, 7 (Nov. 1984), 635-652.

In this article, Bruffee covers a rationale for collaborative learning, the relationship of that rationale to classroom practice, and implications. His discussion also centers on discourse communities; he sees collaborative learning as providing for particular kinds of conversation, social contexts for that conversation, and communities. His preoccupation with conversation is relevant; he states that "writing always has its roots deep in the acquired ability to carry on the social symbiotic exchange we call conversation." Supplementary reading.

Brusaw76

Brusaw, C. T., Alred, G. J., and Oliu, W. E. *The Business Writer's Handbook*. New York: St. Martin's Press, 1976.

This style guide is a practical reference book that provides examples. However, it has some of the limitations of Strunk and White. For further information, see the annotations for [Strunk79] and [Williams89].

Chicago82

The Chicago Manual of Style. Chicago: University of Chicago Press, 1982.

This manual is one of the most complete handbooks available. It contains information appropriate for people who publish documents as well as those who write and edit them.

Christensen78

Christensen, Francis, and Christensen, Bonniejean. *Notes Toward a New Rhetoric: Nine Essays for Teachers*. New York: Harper & Row, 1978.

In the preface, the authors point out that there is no evidence for a correlation between knowledge of grammar and writing ability. Learning elements of style by reading great works is equally problematic. The authors propose, instead, a "generative rhetoric" of the sentence based on levels of structure where the student "adds further levels to what he has already produced, so that structure itself becomes an aid to discovery." A generative rhetoric of the paragraph applies the principles used in analyzing the sentences and sees the paragraph as a macrosentence. The two chapters on sentence and paragraph structure will be of most interest.

Curtis88

Curtis, B., Krasner, H., and Iscoe, N. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM 31*, 1 (Nov. 1988), 1268-1287.

Abstract: The problems of designing large software systems were studied through interviewing personnel from 17 large projects. A layered behavioral model is used to analyze how three of these problems—the thin spread of application domain knowledge, fluctuating and conflicting requirements, and communication bottlenecks and breakdowns—affected software productivity and quality through their impact on cognitive, social, and organizational processes.

The study described in this article found that three characteristics set exceptional software designers apart from their colleagues. One of these characteristics was strong communication skills. The others were familiarity with the application domain and becoming "consumed with the performance of their projects."

DohenyFarina86

DohenyFarina, Stephen. "Writing in an Emerging Organization: An Ethnographic Study." *Written Communication 3*, 2 (Apr. 1986), 158-185.

Abstract: This study explored the collaborative writing processes of a group of computer software company executives. In particular, the study focused on the yearlong process that led to the writing of a vital company document. Research methods used included participant/ observations, open-ended interviews, and Discourse Based Interviews. A detailed analysis of the executive collaborative process posits a model that

describes the reciprocal relationship between writing and the organizational context. The study shows the following: (1) how the organizational context influences (a) writers' conceptions of their rhetorical situations, and (b) their collaborative writing behavior; and (2) how the rhetorical activities influence the structure of the organization.

This article is interesting because of its subject matter and the scene it describes. DohenyFarina focuses on a double interaction: how social and organizational contexts affect the writing of a business plan and how the writing of that plan affects the organization. He describes, for example, how the writing situation prompts and exposes tension between promotional visions and clear production plans. The detailed introduction, outlining of theoretical assumptions, and procedures section will be of interest to writing researchers; the body of the article will be more relevant for software engineers. Recommended for the instructor and students.

Duffy81

Duffy, Thomas M. "Organising and Utilizing Document Design Options." *Information Design Journal, Ltd. 2* (1981), 256-266.

Abstract: In this discussion paper, the author concentrates on the problems of modeling the design process as a means of closing the gap between research and practice in information design. He proposes a new document design model but notes that competing objectives, in particular cost constraints, may prevent the implementation of good design procedures in practice.

Duffy proposes a systems analysis model of document design. Although he focuses on instructional text, much of his article is broadly applicable; and the model provides a helpful checklist.

Duffy85

Duffy, Thomas M. *Readability Formulas: What is the Use*? CDC Tech. Rep. 23, Carnegie Mellon University, Nov. 1985. Also appears as a chapter in Duffy, T. M., and Waller, Robert. *Designing Usable Text.* Orlando, Fla.: Academic Press, 1985.

Duffy identifies factors that must be taken into account when determining the readability of a document. These include format, graphics, and the reader's subject matter knowledge and reading skill.

Dunkle88

Dunkle, Susan B., and Pesante, Linda Hutz. "The Role of the Writer on the Software Team." *Proceedings of the 35th ITCC,* May 1988, WE51-53.

Abstract: The growth of the computer field has been a major factor in the growth of technical writing as a profession. Software developers are beginning to recognize the need for technical writers at all stages of the software life cycle from the development of requirements to the implementation of the system. This paper explores the areas of commonality between the technical writing process and software development process and the special talents that technical writers bring to a software development team.

This paper stresses the contributions that technical writers can make throughout the software life cycle. It is recommended reading for students. If they become aware of the range of skills technical writers have, they will be able to work more productively with technical writers.

Felker81

Felker, D., Pickering, Frances, Charrow, Veda R., Holland, V. Melissa, and Redish, Janice C. *Guidelines For Document Designers.* Washington, D.C.: American Institutes for Research, 1981.

This book presents guidelines for 25 principles concerning text organization, writing sentences, typography, and graphics. With each guideline are: explanations, examples, advice, and a short summary of relevant research. The research cited on comprehension, recall, etc., provides reminders that principles are important in relation to the activities that readers perform, not in and of themselves. The text can serve as a desk reference if the reader first becomes familiar with the contents.

Flower81

Flower, Linda, and Hayes, John R. “A Cognitive Process Theory of Writing.” *College Composition and Communication 32* (Dec. 1981), 365-387.

Based on their research with writers performing think-aloud protocols, the authors introduce their cognitive process theory, which sees composing as a goal-directed, hierarchical thinking process. This article will provide the instructor with an understanding of the issues that are fundamental to a cognitive approach to writing.

Flower89

Flower, Linda. *Problem-Solving Strategies for Writing*. San Diego, Calif.: Harcourt Brace Jovanovich, 1989.

Although this book is clearly geared to undergraduates, it contains a great deal of useful information. Chapter headings include: "Understanding Your Own Writing Process," "Making Plans," and "Organizing Ideas." There are two chapters on audience and two on revising and editing. Instructors who have never taught writing should read this book; small sections will be appropriate for even sophisticated students.

Guillemette87

Guillemette, Ronald A. "Prototyping: An Alternate Method for Developing Documentation." *Technical Communication 34*, 3 (Aug. 1987), 135-141.

Abstract: Documentation can be developed more effectively with a prototyping approach, says the author, who first explains the techniques and benefits of system prototyping and then shows how the method can be applied to documentation. [From the introduction.]

The author considers the limitations of the linear prespecification approach and argues for attending to the iterative nature of the document design process through reader evaluation and comments in the revision cycle. This is an interesting article, but it stresses the "what" more than the "how." An engineer reading this might struggle with the application of rapid prototyping to writing. Guillemette doesn't specify how needs analysis for software is like needs analysis for documentation.

Guinan87

Guinan, Patricia J. and Bostrom, Robert P. *Communication Behaviors of Highly-Rated vs. Lowly-Rated System Developers: A Field Experiment*. IRMIS Working Paper #W707, Institute for Research on the Management of Information Systems, Indiana University, Bloomington/Indianapolis, Ind., May 1987.

Abstract: Communication is often touted as the solution for many organizational problems, but there is little empirical evidence to back this claim. This study was designed to demonstrate the importance of specific communication behaviors to organizations that develop computer-based information systems (CIS). The ability of CIS developers to effectively communicate while eliciting system requirements from CIS users is often cited as a critical factor for successful CIS development. The importance of communication in the developer-user relationship is clearly recognized and documented.

However, empirical research that examines the extent to which it is important and/or identifies specific communication behaviors that make an effective communicator is virtually nonexistent.

An empirical investigation was conducted in a field setting with 55 systems developers who had been related as demonstrating high or low effectiveness by their supervisors. The investigation revealed that while eliciting system requirements from a user, highly-rated developers used specific communication behaviors more frequently than their more lowly-rated counterparts. Moreover, the specified communication behaviors provided substantial discrimination between the two groups. The implications of these results are discussed and future research directions outlined.

This paper focuses primarily on oral communication skills. However, it contains some information on writing, along with a useful bibliography.

Halloran78

Halloran, S. Michael. "Technical Writing and the Rhetoric of Science." *Journal of Technical Writing and Communication 8*, 2 (1978), 77-88.

Abstract: The traditional view of rhetoric and science as sharply distinct has helped reduce the technical writing course to mere vocational training. Current thinking in rhetorical theory and philosophy of science supports the contrasting view that science is rhetorical. Salient aspects of the rhetoric of science are illustrated by Crick and Watson's discovery of the structure of DNA, as recorded in Watson's "The Double Helix"[1]. Analysis of the rhetoric of science suggests that the study of technical writing could be central to liberal education for a technological society.

This is interesting supplementary reading for the instructor, but it is not essential.

Hamlet86

Hamlet, Richard. "A Disciplined Text Environment." *Proceedings of the International Conference University of Nottingham,* Apr. 1986, 78-89. This article also appears, under the same title, in *Text Processing and Document Manipulation,* J. C. van Vliet, ed., Cambridge: Cambridge University Press, 1986.

Abstract: Computer text processing is still in the assembly-language era, to use an analogy to program development. The low-level tools available have sufficient power, but control is lacking. The result is that documents produced with computer assistance are often of lower quality than those produced by hand: they look beautiful, but the content and organization suffer. Two promising ideas for correcting this situation

are explored: (1) adapting methods of modern, high-level program development (stepwise refinement and iterative enhancement) to document preparation; (2) using a writing environment controlled by a rule-based editor, in which structure is enforced and mistakes more difficult to make.

On the topic of rules, this is an interesting article to juxtapose with [Miller80]; for example, in considering writing environments controlled by rule-based editors, one might want to argue that educating the practitioner is preferable, in the long run, to controlling the method. Hamlet makes strong and useful analogies between creating a document and developing a program. He pairs, for example, formatting programs with assemblers, and the current word processing situation, with the "undisciplined" use of programming languages that preceded "modern programming practice." The brief discussions of stepwise refinement and iterative enhancement are insightful and on target. This is essential reading for instructors and is appropriate for students too.

Hartman89

Hartman, Janet D. "Writing to Learn and Communicate in a Data Structures Course." *ACM SIGCSE Bulletin 21*, 1 (Feb. 1989), 32-36.

Hartman's writing-across-the-curriculum experience has been precisely applied to the data structures classroom, but her writing activities can certainly be extended to fit in any computer science course. She uses four types of microthemes (short essays on 5x8 index cards): summary, support for a thesis, generating a thesis from provided data, and quandary posing. Other assignments involve varying the contexts for writing so that students deal with audience issues by assuming roles such as expert or novice, or with issues of genre, by writing in a number of formats—one paragraph response, memo, report, etc. An excellent short article that provides concrete ways of making cost effective changes in writing assignments and handling evaluation. Essential for instructors.

Hayes87

Hayes, John R., Flower, L., Schriver, K., Stratman, J., and Carey, L. "Cognitive Processes in Revision." *Advances in Applied Psycholinguistics: Reading, Writing and Language Processing, Vol. 2*, S. Rosenberg, ed. Cambridge University Press, 1987, 176-240.

This is a comprehensive and lengthy treatment of revision which also includes a short literature review, the [Flower81] process model of composing, and even an introductory argument on the value of using think-aloud protocols in theory building and testing. Readers should be prepared to speed up and slow down depending on interest in the

individual subtopics. Each of the major subprocesses in the revision model (task definition, evaluation, problem detection, problem diagnosis, and strategy selection) is treated in detail. The chapter concludes with a useful summary of major findings.

Hester81

Hester, S. D., Parnas, D. L., and Utter, D. F. "Using Documentation as a Software Design Medium." *The Bell System Technical Journal 60*, 8 (Oct. 1981), 1941-1977.

Abstract: This article describes a software design method based on the principles of separation of concerns and information hiding. The principle of separation of concerns is used to structure the design documentation, and information hiding is used to guide the internal design of the software. Separation of concerns requires that design information be divided into clearly distinct and relatively independent documents. The design documents are the main products of the initial design phase, and are carefully structured to (i) expose open issues, (ii) express design decisions, and (iii) ensure that information is recorded in a way that allows it to be readily retrieved later. Information hiding is used to design software that is easy to change. We have applied many elements of the design method to the development of the No. 2 Service Evaluation System (SES), a multiprocessor data acquisition and transaction system. Our experiences in applying the design method are described, and some examples are included.

An excellent article. The authors treat the scope, use, and design considerations for the software design associated with each step that the document covers. They also provide sound principles to guide the preparation of software documentation. The authors stress, in very clear terms, the relationship between design and documentation: "Since documentation is the main product of the design phases, it is important and must be produced with the same discipline and care with which code is produced." While they are candid about the costs of adhering to this discipline, they also "feel that the cost of neglecting it is even higher." Recommended for instructors and students.

Hoare84

Hoare, C. A. R. "Programming: Sorcery or Science?" *IEEE Software 1*, 2 (Apr. 1984), 6-16.

Abstract: Professional programming practice should be based on underlying mathematical theories and follow the traditions of better established engineering disciplines. Success will come through

improved education.

In hypothetical terms, Hoare considers three present-day roles of computer programmers. They are: craftsmen who serve apprenticeships and develop skills by experience, high priests who are served by a devoted team of acolytes yet held in awe and fear by the public, and modern engineering professionals. These views of programming as craft, magic, and science can be neatly juxtaposed to Young's treatment of writing as art, craft, gift, and knack in [Young80].

Houp92

Houp, Kenneth W. and Pearsall, Thomas E. *Reporting Technical Information, 7th Edition*. New York: Macmillan Publishing Co. 1992.

This textbook has several strengths: three excellent chapters on document design, a long chapter on oral presentations, and a section on collaboration. In addition, it provides a clear, if basic, discussion of the writing process, checklists for writers, and a helpful set of references. Appropriate for both teachers and students.

Jones88

Jones, Patricia L., and Doyle, Kelly M. "Modularizing Software Documentation." *Proceedings of the 35th ITCC,* May 1988, WE49-50.

Abstract: One of the most frequently faced challenges for technical writers is keeping the manuals current when software changes. This can be compounded by the need to produce different manual versions to accommodate different hardware and operating systems. Carnegie Group Inc. took a major step towards solving this problem by organizing the documentation using the modularity principles developed by software engineers. The documentation was redesigned so that modifications required for different computing environments were greatly reduced, and in some cases automated. The strategy involved reorganizing the content by isolating conceptual information from machine-dependent procedural information, and using conditional facilities of our text formatter to produce different manuals for different environments. The resulting structure and methodology can now be used across all product documentation to make porting and maintenance tasks easier and less time-consuming.

The authors describe in detail their procedures for planning and maintaining a large set of documentation. They discuss topics such as modularity and configuration management, as well as their rationale for the choices they made.

Kernighan74

Kernighan, Brian, and Plauger, P. J. *The Elements of Programming Style.* New York: McGrawHill, 1974.

The authors note that the form and approach of their book has been strongly influenced by *The Elements of Style* by Strunk and White (see [Strunk79]). While we have reservations about the latter's concentration on rules and principles of correctness, we recognize Kernighan and Plauger's contribution in drawing on the similarities between programming and writing.

Kinneavy71

Kinneavy, James L. *A Theory of Discourse: The Aims of Discourse.* Engle wood Cliffs, N. J.: Prentice-Hall, 1971.

A comprehensive text that includes classical and contemporary approaches to teaching writing and speech. The opening chapter provides information on models and theories of communication. Section 3 (on the nature, logic, organization, and style of reference discourse) will be most relevant to software engineers. For the instructor.

Kirkman70

Kirkman, A. J. "The Communication of Technical Thought." *The Engineer and Society*, E. G. Semier, ed. London: Institute of Mechanical Engineers, 1970, 180-185.

Abstract: There is much concern among employers about the poor command of English shown by engineering and science graduates; this is a serious matter when it results in failure to communicate technical information. The Department of English and Liberal Studies in the Welsh College of Advanced Technology has launched an investigation into the problems of scientific communication. Preliminary results show that the faults are by no means all on one side and that schoolteachers must take much of the blame; but barriers to communication are often raised artificially by the scientists themselves.

The author provides a very good introduction to long-standing problems in technical communication; the discussion on sequential and associative types of mind is interesting. Kirkman also details, and dispenses with, a number of "excuses" that are often made for inarticulate scientific writing, including the notion that English grammar is too rigid and the idea that "scientists are not practiced in slowing down their thinking to a rate appropriate to the writing process."

Knuth84

Knuth, Donald E. “Literate Programming.” *The Computer Journal 27*, 2 (1984), 97-111.

Abstract: The author and his associates have been experimenting for the past several years with a programming language and documentation system called Web. This paper presents Web by example, and discusses why the new system appears to be an improvement over previous ones. See [Bentley86].

Knuth88

Knuth, D. E., Larrabee, T., and Roberts, P. M. *Mathematical Writing.* STANCS881193, Stanford University, 1988.

A portion of this report is a minicourse on technical writing. Its value lies in the example Knuth sets in providing writing instruction in a content area.

Kraemer88

Kraemer, Kenneth L., and King, John Leslie. “Computer-Based Systems for Cooperative Work and Group Decision Making.” *ACM Computing Surveys 20*, 2 (June 1988), 115-146.

Abstract: Applications of computer and communications technology to cooperative work and group decision making has grown out of three traditions: computer-based communications, computer-based information service provision, and computer-based decision support. This paper reviews the group decision support systems (GDSSs) that have been configured to meet the needs of groups at work, and evaluates the experience to date with such systems. Progress with GDSSs has proved to be slower than originally anticipated because of shortcomings with available technology, poor integration of the various components of the computing “package,” and incomplete understanding of the nature of group decision making. Nevertheless, the field shows considerable promise with respect to the creation of tools to aid in group decision making and the development of sophisticated means of studying the dynamics of decision making in groups.

This article discusses GDSSs (group decision support systems) and new variants which aid group collaboration on common tasks such as: “setting meetings, sharing information, outlining ideas, and evaluating proposals.” Many of these capabilities currently exist for individuals but have not yet been adapted for group activity. Specifically of interest is The Collaboration Laboratory, which “focuses on writing and argumentation and involves verbal models and qualitative techniques through the manipulation of text-oriented data and graphical images.” Includes text-oriented tools: a “common human-machine interface,

WYSIWIS (what you see is what I see) for presentation of images of shared information for all participants, public (shared) and private (not shared) windows on the workstations, and applications such as a group method of preparing outlines of ideas and associated text and a group method of evaluating plans and programs that have already been developed." The method of outlining resembles ThinkTank but includes collaboration capabilities. Another GDSS, The Group Network, focuses on "interactive computer support for small groups in geographically dispersed but nearby locations such as offices within a building." Participants are able to create, edit, or simply exchange graphics, text, or numbers although only one person at a time can do so.

Lehman86

Lehman, John A., "Program Design and Rhetoric." *IEEE Software* (May 1986), 71-73.

Lehman argues that programmers and managers who have looked to engineering and mathematics as disciplinary guides for the development of program design have neglected to consider another very relevant discipline—rhetoric. "From an information processing point of view, the problems faced in rhetoric are very similar to those faced in programming, and the rhetorical solutions to these problems resemble the major tenets of structured programming and structured design." We agree with Lehman but find that his emphasis on "structure" limits his definition of rhetoric to arrangement and style. In doing so, he fails to account for "invention" or problem definition and analysis in rhetoric and writing. Essential reading for the instructor, and recommended for students.

Levine91

Levine, Linda, Pesante, Linda H., and Dunkle, Susan B. "Implementing the Writing Plan: Heuristics from Software Development," *The Technical Writing Teacher 28*, 2 (Spring 1991), 116-125.

Abstract: In considering approaches to software development and software life cycle models, we have gained insight into the problems of writers and the teaching of technical communication. In particular, this article focuses on analogies between the two processes and domains and suggests concrete ways for writers to proceed beyond analysis. The field of software engineering has proven to be a rich source of strategies for implementing the plan—for navigating the crucial area between planning and revising that is conventionally and generically called writing.

In addition to elaborating on some of the information provided in this curriculum module, the article describes students' experiences with writing in Carnegie Mellon's Master of Software Engineering program.

Mattes76

Mathes, J. C., and Stevenson, D. W. *Designing Technical Reports: Writing for Audiences and Organizations*. New York: Bobbs-Merrill Educational Publishing, 1976.

This textbook offers a good treatment of purpose/problem statements in an organizational setting. It ties the issues of complex audience and component structure together, emphasizing that particular readers read particular parts of reports. Writers, therefore, should think about reports in terms of opening and discussion components. (For contrary findings, see [Spilka88].) They should also design reports that move from general to particular between the two components, and make the components self-sufficient. Note: the matrix for audience analysis may be too complex for easy use.

McLeod88

McLeod, Susan H. "The Portfolio Method For Teaching Technical Communication." *Technical Communication 35*, 3 (1988), 238-239.

Two pages of good, concrete, teaching strategies in the portfolio method. McLeod provides smart, efficient suggestions on spot-grading and the excellent instruction sheet that she gives her students.

Meinke87

Meinke, John G. "Augmenting a Software Engineering Projects Course with Oral and Written Communications." *ACM SIGCSE Bulletin 19*, 1 (Feb. 1987), 238-243.

This paper describes a "Senior Projects" course and provides information about the two oral presentations and seven formal written reports that are required. Writing assignments include: extended abstract, justification report, milestones, requirements, system documentation, user manual, and final system report. The article concentrates on content requirements as opposed to teaching method (i.e., what approach is taken to technical communication in the classroom? and how have the students benefited from that approach?), but it is a useful example of the integration of technical communication and software engineering in the classroom. Recommended for instructors.

Meyer82

Meyer, Bonnie J. F. "Reading Research and the Composition Teacher: The Importance of Plans." *College Composition and Communication 33*, 1 (Feb. 1982), 37-49.

Meyer summarizes research related to planning and discusses three functions of writing plans. The topical function helps the writer generate and organize main ideas; the highlighting function helps the writer show priorities and important relations between ideas. The informing function helps writers decide "how to present new knowledge while keeping readers aware of the old." Meyer presents empirical evidence for five basic writing plans that have an impact on readers' comprehension. These plans are: antecedent/consequent, comparison, description, response, and time order. Recommended reading for instructors.

Miller79

Miller, Carolyn. "A Humanistic Rationale for Technical Writing." *College English 40*, 6 (Feb. 1979), 610-617.

Miller considers problematic and lingering assumptions about language and technical writing as the result of a pervasive positivist view of science. While one might be tempted to consider this philosophical piece as secondary reading, it is a concise and thought-provoking critique of "what has been called 'the windowpane theory of language': the notion that language provides a view out onto the real world, a view which may be clear or obfuscated." Miller also considers new directions in the philosophy of science and corresponding new relations between rhetoric and science.

Miller80

Miller, Carolyn. "Rules, Context and Technical Communication." *Journal of Technical Writing and Communication 10*, 2 (1980), 149-180.

Abstract: The concept of "rule" derived from linguistics and anthropology, provides a way of understanding the relationship between context, purpose, and message production and interpretation. "Rules" are shared expectations which structure situations and guide individual action. This paper shows some of the concepts that have come out of rules theory in communication research and suggests their particular relevance and utility to understanding the problems and situations in technical

communication.

This is a thoughtful article that places "the source of authority for rules in those who use them, not in some impersonal or absolute authority." Miller's discussion on context as a hierarchy is particularly interesting, as are the distinctions she makes between constitutive and regulative rules. Constitutive rules cover all permissible game moves while regulative rules govern efficient play.

Mingione83

Mingione, Al. "Iteration, Key to Useful Documentation." *Journal of Systems Management 34* (Jan. 1983), 23-25.

Mingione claims that since iteration is "a principal concept within the development," it is a more "natural" and "easier way to document." We agree with Mingione's approach but find that he skirts the issue of how one produces iterative documentation. It is unrealistic to maintain that through "iteration these drafts will evolve to a communication understood by all" without attending to guidelines, goals, constraints, and planning and outlining techniques, etc. In Mingione's limited sense, iteration means drafts, and perhaps not necessarily improved drafts. His understanding of technical writers is also reactionary—they are largely editors/stylists who don't take part in the development process, but who may "make others interact effectively through documentation." Mingione supports the iterative approach, or writing within the development process, but ironically, not for writers. Essential reading for the instructor and students.

Odell85

Odell, Lee, and Goswami, Dixie. *Writing in Nonacademic Settings.* New York: The Guilford Press, 1985.

This is secondary reading for the instructor. Of special interest are: Colomb and Williams' treatment of form in "Perceiving Structure in Professional Prose" and Miller and Selzer's article, "Special Topics of Argument in [Transportation] Engineering Reports." Two articles addressing the influence of new technologies on composing, Halpern's "An Electronic Odyssey" and Murray's "Composition as Conversation: The Computer Terminal as Medium of Communication," offer insights. The latter articles are already somewhat dated in terms of technologies, but the observations on electronic discourse in general are still relevant. Finally, issues of context are addressed in two articles representing the social approach to technical writing: Faigley's "Nonacademic Writing: The Social Perspective" and Odell's "Beyond the Text: Relations Between Writing and Social Context."

Olsen91

Olsen, Leslie A., and Huckin, Thomas N. *Technical Writing and Professional Communication*. New York: McGrawHill, 1991.

This textbook addresses business, scientific, and technical writing. Two versions are available: for native and for non-native speakers of English. The first chapter, "Why Study Technical Communication," contains convincing information drawn from studies by the American Society for Engineering Education. A section on readability provides concrete advice based on psycholinguistic principles. The authors also present strategies for generating ideas, constructing arguments, stating problems, and testing documents. There are two chapters on visual elements and one each on oral presentations, meetings, and computer documentation. The book contains a list of references and additional reading for each chapter, realistic examples and exercises, and a punctuation guide. Additional information about the language is in the version for non-native speakers. If a single textbook is to be used, this is the one. It is essential reading for instructors and students.

Ong82

Ong, Walter J. *Orality and Literacy: The Technologizing of the Word*. London: Methuen, 1982.

Two chapters in this fascinating book are the most relevant: "Writing restructures consciousness" and "Print, space and closure." Ong points out that many current objections and fears about the inhumanity of computers parallel Plato's objections about writing: that it is artificial, unresponsive, and weakens the mind by destroying memory. Ong's discussion on post-typography and its consequences is also interesting. He explains how "electronic transformation of verbal expression has both deepened the commitment of the word to space initiated by writing and intensified by print and has bought consciousness to a new age of secondary orality." Secondary reading for instructors.

Penrose88

Penrose, John M., and Seiford, Lawrence M. "Microcomputer Users' Preferences For Software Documentation: An Analysis." *J. Technical Writing and Communication 18*, 4 (1988), 355-366.

Abstract: Fundamental requirements for good user documentation have not changed over the years. Manuals must be complete, accurate, clear, readable, and available on time. What has changed are tolerances and standards. Today's users—typically business professionals and even expert technicians and engineers—will no longer accept unreadable and inaccessible publications. The days of documentation with poor aesthetics have passed. This article analyses users' opinions and preferences for microcomputer software documentation. The results

provide valuable guidance for software authors, designers, and publishers.

The results of this survey are generally informative but clearly should not be used to replace audience and task analyses and testing for particular software documentation. Self-reporting is sometimes an inaccurate indicator of how individuals use documentation. The article also addresses the subject of emerging standards for software documentation; it provides the set of 12 sections for comprising a document from the IEEE Working Draft, *Standard for Software User Documentation*. The bibliography includes references to guides and handbooks on software documentation.

Perl80

Perl, Sondra. "Understanding Composing." *College Composition and Communication 31*, 4 (Dec. 1980), 363-369.

Perl uses the terms retrospective and projective structuring to capture the backward and forward nature of composing, "the move from sense to words and from words to sense, from inner experience to outer judgment and from judgment back to experience." She describes the writer's process as recursive in the rereading of bits of texts, going back to interpret the topic and the "felt sense" that surrounds that topic. Perl's understanding of the writing process as recursive is somewhat different from Flower and Hayes' who, in addition, stress how routines and subroutines are embedded within the phases of composing [Flower81]. This article is a significant and unusual blend of practical and philosophical research. Secondary reading for instructors.

Perlman86

Perlman, Gary. *Multilingual Programming: Coordinating Programs, User Interfaces, OnLine Help, and Documentation*. TR8605, Wang Institute Graduate Studies, 1986.

Perlman points out that to separate documentation from issues of programming, user interfaces, or online help is wasteful; to have different people, programmers and writers, spending "their time expressing the same idea in different languages" is unnecessary effort. Following in Knuth's tradition of literate programming, multilingual programming extends the process to coordinate program implementation with "the use of parameterized information for domains outside programming, like documentation and user interfaces."

Perlman89

Perlman, Gary. *User Interface Development*. Curriculum Module SEI-CM-17; DTIC: ADA235699, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., November 1989.

Capsule Description: This module covers the issues, information sources, and methods used in the design, implementation, and evaluation of user interfaces, the parts of software systems designed to interact with people. User interface design draws on the experiences of designers, current trends in input/output technology, cognitive psychology, human factors (ergonomics) research, guidelines and standards, and on the feedback from evaluating working systems. User interface implementation applies modern software development techniques to building user interfaces. User interface evaluation can be based on empirical evaluation of working systems or on the predictive evaluation of system design specifications.

Because of the emphasis on users, there are many parallels between the development of user interfaces and the development of documents.

Pesante91

Pesante, Linda Hutz. "Integrating Writing into Computer Science Courses," *SIGCSE Bulletin 23*, 1 (March 1991), 205-209.

Abstract: Writing can and should be an integral part of computer science and software engineering courses. This paper describes an approach to teaching writing that can be used by instructors of technical courses; it suggests both content and teaching techniques. The paper also discusses how to enlist the aid of technical writers and technical writing teachers.

Written as a "how to" paper, this is a succinct guide to teaching writing. It includes a description of strengths that instructors of technical courses and writing experts each bring to the classroom.

Pesante92

Pesante, Linda Hutz. *Applying Software Engineering Skills to Writing.* Videotape TECH-LP-01-01, Technology Series, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1991.

Description: Software engineers spend significant time communicating technical information, yet many find writing difficult. Although writing is hard, even for good writers, it is not a mysterious art. Software engineers can become more effective writers by approaching writing as an engineering task. In Applying Software Engineering Skills to Writing, *Linda Pesante draws the similarity between the software life*

cycle and the writing process. She discusses how approaches such as prototyping and iterative development apply to writing, and describes how to analyze a writing task. She also notes how awareness of individual work styles can be used to the writer's advantage. By applying their software skills to writing, engineers can get results.

This videotape can be used to introduce writing instruction and to convince students that they need such instruction. Students from the Master of Software Engineering program at Carnegie Mellon reinforce key points and comment on the importance of writing. The tape, which can be purchased from the SEI, is 17 minutes, 51 seconds long.

Redish89

Redish, Janice C. "Writing in Organizations." in Myra Kogen, Ed. *Writing in the Business Professions*. Urbana, Ill.: National Council of Teachers of English, 1989, 97-124.

This chapter describes the Document Design Center's process model, which is interesting to consider alongside the waterfall model of the software life cycle. Clearly, both models share phases even though these phases are discussed in different terms. Although the DDC model specifies activities, such as analyzing audience and evaluating, at prescribed stages, Redish notes that "...all the boxes should be connected with arrows. The process of writing is iterative, not linear."

Redish also reports studies that show the importance of writing for people who hold professional, technical, or managerial jobs. Two studies found that these professionals spend about 20 percent of their time writing; in another study, 96 percent of 245 "engineers of distinction" said that good writing skills had helped them advance.

Other topics include features that make information accessible to readers and the reason readability formulas don't work. Highly recommended reading for instructors; recommended reading for students.

Redish87

Redish, Janice C. "Integrating Art and Text." *Proceedings of the 34th ITCC,* May 1987, VC4-7.

Abstract: Art can make a print manual or online tutorial or help screen more interesting. Art can also help readers understanding the message in the text. In this paper, I explore different ways in which art can help readers.

The focus of this article is user documentation, and the advice is basic and concrete.

Rehe81

Rehe, Rolf F. *Typography: how to make it most legible.* Carmel, Ind.: Design Research International, 1981.

Rehe provides a solid introduction to the printed word. He briefly describes research findings and makes recommendations that are appropriate for novices and experts alike. The book is short and easy to read.

Rice84

Rice, Patricia Brisotti, and Dorchak, Susan Fife. "A Course in Documentation and Technical Communication." *ACM SIGCSE Bulletin 16*, 4 (Dec. 1984), 7-8.

Abstract: The Computer Science program at the C. W. Post Campus of Long Island University, which has approximately four-hundred undergraduate majors, is predominantly software oriented. A course in communication is required and taken at the sophomore level. The concepts covered include information gathering, user-friendly programming, system and program documentation, written and verbal presentations. This course also prepares the students for the Management Engineering Master's degree offered at C. W. Post.

The skeletal description touches on the goals, objectives, and content of this class. In fact, one might see this description as an annotated syllabus. The course's commitment to integrating system design and documentation is ambitious; the sections on internal and external documentation are especially interesting and unusual.

Rogers81

Rogers, Everett M. and Kincaid, D. Lawrence. *Communication Networks*. New York: The Free Press [A Division of Macmillan Publishing Co., Inc.], 1981.

In this excellent text, Rogers and Kincaid offer a "new paradigm for research" based on their convergence model of communication and network analysis. Readers particularly interested in the role of communication in technology transfer should also consult Rogers' *Diffusion of Innovations* published by The Free Press in 1983.

Rohman65

Rohman, D. Gordon. "Pre-Writing: The State of Discovery in the Writing Process." *College Composition and Communication 16* (May 1965), 106-112.

Rohman views writing as a linear process. His discussion of good writing, bad writing, and creativity is interesting.

Rose80

Rose, M. "Rigid Rules, Inflexible Plans, and the Stifling of Language: A Cognivist's Analysis of Writer's Block." *College Composition and Communication 31*, 4 (Dec. 1980), 389-401.

Rose's research on the process of composing reveals that writer's block is not so mysterious. It often occurs when writers impose premature restrictions on their use of language.

Roundy85

Roundy, Nancy, and Mair, David. *Strategies for Technical Communication*. Boston, Mass.: Little, Brown, 1985.

This undergraduate textbook takes an especially comprehensive approach to audience analysis; the authors chart positions within (or outside of) the organization, and positions based on "cluster of interest" and egocentrism. There are useful outlines and matrices for analyzing audience characteristics. Writing models are also provided to demonstrate key points.

Schutte83

Schutte, William M., and Steinberg, Erwin R. *Communication in Business and Industry*. New York: Holt, Rinehart and Winston, 1983.

The authors provide standard handbook information and more, including coverage of communication theory, the "climate of business," and special problems in technical and professional writing. This text is less an undergraduate text than most, and is also geared to people in business, industry and government. It contains information about oral communication as well as writing. Although the book is no longer in print, it may be available in libraries, particularly university libraries.

Selzer83

Selzer, Jack. "The Composing Process of an Engineer." *College Composition and Communication 34*, 2 (May 1983), 178-187.

This classic article considers how Kenneth Nelson, (transportation) engineer, generally writes the kinds of documents that are part of an engineering project: qualifications statement, proposal, presentation, progress reports, technical memos, and final report. Nelson's comparatively linear process of composing and his efficient reuse of documentation is of interest. Selzer was not primarily concerned with correspondences between engineering and composing, yet he comments on the level of detail in both Nelson's outlines and his engineering plans. He speculates that planning for documentation comes "naturally to professionals who must plan and coordinate complicated engineering tasks." He also notes that "The Critical Path

Diagram and the Project Task Flow are remarkably analogous to models of the writing process." The article is both informative and easy to read. Essential reading for the instructor and recommended for students.

Shore85

Shore, John. *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety*. New York: Viking Press, 1985.

This entertaining book looks at programming as a literary activity, as mathematics, and as architecture. Shore addresses a wide audience, including technical writers, computer novices who need a gentle introduction, and experts who have forgotten home truths.

Simon81

Simon, Herbert A. *The Sciences of the Artificial*. Cambridge, Mass.: MIT Press, 1981.

Simon's discussion of the science of design is relevant to computer science, software engineering, and communication. "Design," Simon argues, "is concerned with how things ought to be, with devising artifacts to attain goals." In addition, the chapter on "The Psychology of Thinking" shows how the relation between linguistic theories and information-processing theories of thinking continues to grow closer.

Sommers80

Sommers, Nancy. "Revision Strategies of Student Writers and Experienced Adult Writers." *College Composition and Communication 31* (Dec. 1980), 378-388.

In this article, Sommers argues that revision has not received its due and is inadequately represented in linear models of the writing process; the possibility of revision "distinguishes written text from speech." In her study of student and adult writers, Sommers notes that students focus on words or phrases. They lack "heuristics to help them reorder lines of reasoning or ask questions about their purposes and readers." Experienced writers revise at a number of levels to discover and shape meaning. Recommended for the instructor.

Spilka88

Spilka, Rachel. "Studying Writer-Reader Interactions in the Workplace." *The Technical Writing Teacher 13*, 3 (Fall 1988), 208-221.

Abstract: Current models of audience analysis fail to account for writer-reader interactions in the workplace. Spilka builds a case for studying in-depth such interactions, and she describes her use of methodological triangulation in a study of corporate engineers' composing processes.

Her results challenge prevalent assumptions about writing for multiple audiences.

This article is supplementary reading for the instructor and its level of detail may make it most appropriate for those who teach technical writers, not software engineers. Nonetheless, it is an excellent treatment of writing to multiple audiences. Spilka's challenges to commonly held assumptions about audience are most interesting. She discovers, for example, that successful corporate engineers try not to write sections of a document for any one segment of the audience. They also focus on more (not less) knowledgeable readers, adjust their audience analysis throughout, and try not to analyze their audience and choose matching strategies on the sole basis of organizational roles.

Strunk79

Strunk, William, Jr., and White, E. B. *The Elements of Style.* New York: Macmillan, 1979.

This handbook includes rules of usage and principles of communication. One drawback of this book is that the rules of usage are presented in a way that suggests there is clear right and wrong. The principles of communication are more useful, and examples are provided. (See [Williams89] for a distinction between the rules that must be followed and those that are flexible, and [Miller80] for a general discussion of the concept of rules.)

Sullivan88

Sullivan, Sarah L. "How Much Time Do Software Professionals Spend Communicating?" *Computer Personnel 11*, 4 (Sept. 1988), 2-5.

This article can be useful for convincing students that software engineers do not work in isolation, that they need communication skills. Sullivan's focus, however, is primarily on oral communication.

Taylor82

Taylor, Tamara, and Standish, Thomas A. "Initial Thoughts on Rapid Prototyping Techniques." *ACM SIGSOFT Software Engineering Notes 7*, 5 (Dec. 1982), 160-166.

Abstract: This paper sets some context, raises issues, and provides our initial thinking on the characteristics of effective rapid prototyping techniques. After discussing the role rapid prototyping techniques can play in the software lifecycle, the paper looks at possible technical approaches including: heavily parameterized models, reusable software, rapid prototyping languages, prefabrication techniques for system generation, and reconfigurable test harnesses. This paper

concludes that a multifaceted approach to rapid prototyping techniques is needed if we are to address a broad range of applications successfully—no single technical approach suffices for all potentially desirable applications.

Although the authors are concerned with the rapid development of initial versions of a system, rapid prototyping is a useful method for developing initial versions of a document. Supplementary reading.

TechComm91

Society for Technical Communication. *Technical Communication 38,* 4 (1991). Special Issue: Collaborative Writing.

This special issue contains 10 articles on collaboration. Topics include collaborative writing in the workplace, collaboration of writers and their readers, and collaboration between the business and academic communities to provide writing instruction.

TechComm88

Society for Technical Communication. *Technical Communication 35,* 3 (1988).

This issue features three articles on manuals: Gillihan and Herrin's "Evaluating Product Manuals for Increased Usability," Huston and Southard's "Organization: The Essential Element in Producing Usable Software Manuals," and Southard's "Practical Considerations in Formatting Manuals." Each article is an introduction to the topic, and the three together are useful for the reader trying to gain a general understanding of issues related to writing manuals. The level of detail is not sufficient for one to directly apply this information to a particular document, but the references are useful pointers to more specific discussion.

Tichy88

Tichy, Henrietta J. *Effective Writing For Engineers, Managers, Scientists.* New York: John Wiley, 1988.

This excellent book is more a series of articles than a textbook. Tichy provides common-sense advice that is appropriate reading for students and teachers. The book also contains sections on usage that can be used as a handbook. Tichy is a pragmatist who attends to stylistic issues without holding to rules at all costs.

Tufte83

Tufte, Edward R. *The Visual Display of Quantitative Information.* Cheshire, Conn.: Graphics Press, 1983.

Tufte provides comprehensive coverage of visual display, with an emphasis on graphs and charts. He stresses the need to select the most appropriate form for communicating data, a form that is complete and clear, and is not misleading. Design principles are presented in this context.

vande Kopple82

vande Kopple, William J. "Functional Sentence Perspective, Composing, and Reading." *College Composition and Communication 33* (Feb. 1982).

This article provides good background information for the instructor; it is not appropriate for students.

Walton87

Walton, Richard E., and Balestri, Diane. "Writing as a Design Discipline: Exploring the Relationship Between Composition and Programming." *Machine Mediated Learning 2,* 1, 2 (1987), 47-65.

Abstract: Many of the difficulties college freshmen encounter as they write stem from their misconceptions about the nature of writing. In this paper we suggest that linking instruction in computer programming and composition offers students a new way to understand and to practice writing as a design discipline. The process of structured computer programming is much like the process of writing purposive prose. Both are processes of design: of determining purpose and audience, of problem-solving, structuring, refining and drafting. We describe the ways in which this insight was applied with positive results to the teaching of writing at two different institutions, the University of Montana and Bryn Mawr College. The computer is a powerful tool for interdisciplinary learning and practice of design, but it is not an effective tool unless used in the context of good instruction.

The authors' insights on writing and programming as problem-solving activities are valuable but, we believe, somewhat limited. By concentrating on the analogy between writing (a document) and writing a program, Walton and Balestri miss out on larger correspondences between document and software development. For example, in their shared design process (Requirements, Design, Draft, Execute) there is no mention of testing or verification, a crucial procedure, for example, in writing manuals. Even in writing expository prose (the subject matter of this article), the (informal) review process

and the modeling of reader behavior are valuable. The authors' discussion of the Warnier-Orr diagram as an effective and hierarchical substitute for the traditional outline is interesting and also worth juxtaposing with issue trees [Flower89].

Watzman87

Watzman, Suzanne. "Visual Literacy and Document Productivity." *Proceedings of the 34th International Technical Communication Conference*, May 1987, ATA48-49.

Abstract: The power of electronic publishing has put its users, the producers of communications materials, in grave danger of overlooking quality. And those who receive this material are in even greater danger—of missing the message. Being sensitive to quality ensures that electronic publishing technology is an asset to you or your organization. In an environment overloaded with a vast quantity of information, increasing information quality becomes crucial. The new technological tools we have access to are valuable only if applied appropriately. Effective communications materials require a unique combination of technological tools, content, and design to meet their objectives. Information design supplies the essentials for you to help readers get more out of your communications.

This article emphasizes simplicity and suggests "visual structuring techniques" that make information easily accessible to the reader.

Weizenbaum88

Weizenbaum, Joseph. "The Computer is a Mythconstrued Machine." *Technology Review 91*, 8 (Nov. 1988), 2-4.

A short but philosophical piece which, on several occasions, raises the issue of literacy by way of considering myths about computers. Of interest is the discussion on the computer as merely a tool since writing is frequently perceived in the same neutral manner. This article should prompt discussion on tools, ethics, and what Weizenbaum calls the "principal end use" of work.

White86

White, K. B. and Leifer, R. "Information Systems Development Success: Perspectives from Project Team Participants," *MIS Quarterly 10*, 3 (Sept. 1986), 219-225.

This article provides more support for the fact that software developers need strong communication skills in addition to their technical skills.

Williams89

Williams, Joseph M. *Style: Ten Lessons in Clarity and Grace*. Glenview, Illinois: Scott, Foresman and Company, 1989.

This sophisticated but understandable style guide is recommended reading for teachers and students alike; it is practical and provides good examples. And Williams does more than provide guidelines; he explains the underlying principles and gives advice about when to follow and when to ignore a rule. The book ends with a section called "Reasonable Punctuation."

Wright83

Wright, Patricia. "Manual Dexterity: A User-Oriented Approach To Creating Computer Documentation." *Human Factors in Computing Systems*, CHI '83 Conference Proceedings, Dec. 1983, 12-15.

Abstract: This paper will not advocate a list of firm recommendations about document design because it is recognized that design decisions will vary with many factors. Instead, the present discussion will emphasize that when making these decisions it is necessary for designers to take account of how readers will use the information provided. In order to help them do this, a simple framework is proposed which outlines the rudiments of how people interact with technical documents. The advantages of this framework will be illustrated by using it to motivate design decisions at two decision levels. At a "macro" level the document designer must make broad decisions about the contents and format of the manual. At a "micro" level the designer must select particular combinations of linguistic, graphic and typographic options which will help readers locate, understand, and implement the information given in the manual.

Wright identifies three common reader activities—searching, understanding, and applying—and considers the implications of these activities for document design. For example, she discusses how the reader's deliberate choice not to read or the reader's practice of leafing through a document should prompt document designers to appreciate the "search component" and provide better "access structures." An excellent article on the connections between testing, usability, and design. The bibliography provides follow-on readings. Essential for instructor and students.

Also published as a special issue of the *Special Interest Group on Computer and Human Interaction (SIGCHI) Bulletin*, ACM, 1983.

Young70

Young, Richard E., Becker, Alton L., and Pike, Kenneth L. *Rhetoric: Discovery and Change*. New York: Harcourt, Brace, 1970.

This unusual text reflects the interests of its authors, who come from three disciplines: rhetoric, anthropology, and linguistics. The discussions on inquiry (problem solving) and interpretation (shared expectations) reinforce the cognitive and cultural elements of language use. The book is known especially for its "tagmemic grid," a heuristic based on viewing experiences as particles, waves, and fields.

Young80

Young, Richard E. "Arts, Crafts, Gifts, and Knacks: Some Disharmonies in the New Rhetoric." *Visible Language 14*, 4 (1980), 341-350.

This paper also appears in Reinventing the Rhetorical Tradition, ed. A. Freedman and I. Pringle, L&S Books for the Canadian *Council* of Teachers of English, 1980. A version is available as "Concepts of Art and the Teaching of Writing" in The Rhetorical Tradition and Modern Writing, ed. J. J. Murphy, New York: Modern Language Association of America, 1982.

Young addresses the limitations, especially pedagogical, that come from seeing writing as mechanical/grammatical or magical. This is an interesting article to read with Hoare's "Programming: Sorcery or Science?" [Hoare84]. Both Hoare and Young discuss the craftlike, magical, and scientific properties of these two activities—programming and writing.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release
Distribution Unlimited

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
SEI-CM-23

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Software Engineering Institute

6b. OFFICE SYMBOL (if applicable)
SEI

7a. NAME OF MONITORING ORGANIZATION
SEI Joint Program Office

6c. ADDRESS (city, state, and zip code)
Carnegie Mellon University
Pittsburgh PA 15213

7b. ADDRESS (city, state, and zip code)
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

8a. NAME OFFUNDING/SPONSORING ORGANIZATION
SEI Joint Program Office

8b. OFFICE SYMBOL (if applicable)
ESC/ENS

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F1962890C0003

8c. ADDRESS (city, state, and zip code))
Carnegie Mellon University
Pittsburgh PA 15213

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.
63756E	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Technical Writing for Software Engineers {Insert title line 2} {Insert title line 3} {Insert title line 4}

12. PERSONAL AUTHOR(S)
Linda Levine Linda H. Pesante Susan B. Dunkle

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (year, month, day)	15. PAGE COUNT
Final	FROM TO	November 1991	75

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (continue on reverse if necessary and identify by block number)

This module, which was written specifically for software engineers, discusses writing in the context of software engineering. Its focus is on the basic problem-solving activities that underlie effective writing, many of which are similar to those underlying software development. The module draws on related work in a number of disciplines, including rhetorical theory, discourse analysis, linguistics, and document design. It suggests techniques for becoming an effective writer and offers criteria for evaluating writing.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified, Unlimited Distribution

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (include area code)	22c. OFFICE SYMBOL
Thomas R. Miller, Lt Col, USAF	(412) 268-7631	ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19